University of the Philippines Manila
College of Arts and Sciences
Department of Physical Sciences and Mathematics

Visualization of Multivariate Health Data using Self-Organizing Maps

A Special Problem in Partial Fulfillment of
the Requirements for the Degree of
Bachelor of Science in Computer Science

by

Mark Lester Y. Ghany

April 2012

ACCEPTANCE SHEET

The Special Problem entitled **"Visualization of Multivariate Health Data Using Self-Organizing Maps"** prepared and submitted by Mark Lester Y. Ghany in partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science has been examined and is recommended for acceptance.

Geoffrey A. Solano, M.S.
Adviser

EXAMINERS:

	Approved	**Disapproved**
1. Gregorio B. Baes, Ph.D. (candidate)	_____	_____
2. Avegail D. Carpio, M.S.	_____	_____
3. Richard Bryann L. Chua, M.S.	_____	_____
4. Aldrich Colin K. Co, M.S. (candidate)	_____	_____
5. Ma. Sheila A. Magboo, M.S.	_____	_____
6. Vincent Peter C. Magboo, M.D., M.S.	_____	_____
7. Bernie B. Terrado, M.S. (candidate)	_____	_____

Accepted and approved as partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science.

Avegail D. Carpio, M.S.
Unit Head
Mathematical and Computing Sciences Unit
Department of Physical Sciences
and Mathematics

Marcelina B. Lirazan, Ph.D.
Chair
Department of Physical Sciences
and Mathematics

Reynaldo H. Imperial, Ph.D.
Dean
College of Arts and Sciences

Abstract

Data that are multivariate in nature is a type of data that may contain subtle patterns. However, it is considered to be an obstacle in research most of the time since classical statistics may find it encumbering to analyze. However, computational statistics, a collaboration between computer science and statistics, offers a suite of algorithms that may be used to surpass obstacles such as this. The Self-Organizing Map and Data Visualization are examples of these. The Self-Organizing Map is an artificial neural network that employs a process to reduce multidimensional data into a low-dimensional representation while Data Visualization is a process that aims to give the human brain a visual representation of knowledge about certain data. SOM Visualize is a software that makes use of both processes. The tool enables users to input data and visualize several patterns such as clusters, associations, as well as a geographical representation that exist in the data. It may give several hypotheses that may be confirmed through other statistical tests and SOM Visualize has therefore enabled the possibility of analysis of multivariate data.

Keywords: Self-Organizing Map, Data Visualization, Artificial Neural Networks, Multivariate Data, Computational Statistics

Contents

I. Introduction

A. Background of the Study

The advent of database systems has tremendously increased the amount of data that are made available for analysis. Through automation, data have been properly indexed and hence, are easily retrieved through querying. However, because database systems entail an easier way of gathering and storing data, most of these data became large in dimension while containing subtle patterns. Because of these characteristics, it may be encumbering for classical statistics to interpret and analyze these data alone. Thus, a collaboration between the fields of statistics and artificial intelligence was made and the concept of data mining and knowledge discovery was born.

Knowledge Discovery and Data Mining are concepts in computer science that aim to search large datasets for patterns that may be considered as knowledge about the data [1]. These patterns may include anomalies, associations, clusters, and classes and to effectively present these knowledge, one of the tasks included in data mining is Data Visualization - giving the human brain a visual representation of the data [2].

A Self-Organizing Map (SOM) is a type of artificial neural network that is capable of discovering patterns in datasets. It is trained using unsupervised learning and employs a data compression technique known as vector quantization to reduce multi-dimensional data to a low-dimensional representation [3].

Visualizing a SOM may be considered as a post-processing step. A common algorithm used to visualize the lattice is the Unified Distance Matrix (U-matrix) - a technique which delivers a "landscape" of distance relationships of the input data in the data space [4]. Applying a clustering algorithm to a SOM is also a common choice in visualizing defined clusters in the data space [5]. For associations, we use component planes to see relationships between different variables. For other knowledge, further methods such as threshold classification may be applied to the map. And in cases where applicable, SOM and GIS may work hand in hand to present a better visualization of spatial data [6].

Geographic Information Systems (GIS) are systems designed to capture, store, manipulate, analyze, manage, and present all types of geographically relevant data [7]. In 2005,

Google has released an application programming interface (API) for users to be able to embed Google Maps in their own websites and overlay their own data for free. The advent of GIS opened a lot of opportunities to visualize data that are geographic in nature.

The Regional Health Data obtained through surveys conducted by the Philippine Census is an example of a multidimensional data. Health is defined by the World Health Organization as "a state of complete physical, mental, and social well-being and not merely the absence of disease or infirmity [8]." It is considered as a complex system because several factors influence health including social, economic, and environmental variables that may differ per region. Data used in this study include thirty health and demographic variables in categories such as Fertility, Family Planning, Maternal Health, Childhood Mortality, Children's Health and Nutrition, HIV/AIDS - related Knowledge and Behavior, and Violence Against Women. The data is available through the NSO Website (http://census.gov.ph).

B. Statement of the Problem

The Philippine Regional Health Data is available for public information and research purposes. However, to effectively obtain knowledge from these data for future statistical analyses, visualization was sought. Hence, there is a need for a tool to aid in analysis by:

1. Organizing the data using the SOM's algorithm.
2. Visualizing the data using different lattice representations.
3. Creating a geographic map representation of values in data obtained through the lattice visualizations.

C. Objectives

The study aims to develop a tool that would allow for a way to visualize multivariate data. The tool is able to:

1. Create or Load a Visualization (.vis) file - the tool's own file format.
2. Obtain an input data by asking the user to:

(a) Import an excel file containing the data that conforms to a specific data format; or

(b) Enter data through an input facility provided by the tool.

3. Construct a self-organizing map by asking the user to provide:

   (a) The width and height of the map.

   (b) The initial and final learning rate.

   (c) The number of iterations for training.

   (d) The neighborhood decay function.

   (e) The learning rate decay function.

   (f) The node influence function.

4. Train the data set by using the self-organizing map's learning algorithm.

5. Give the user a visualization of potential clusters of data using the unified distance matrix by asking the user the neighborhood size.

6. Give the user a visualization of clusters through the k-means clustering algorithm by asking the user the number of desired clusters.

7. Give the user a visualization of each variable to be able to assess potential associations.

8. Give the user a GIS visualization of:

   (a) Geographic clusters.

   (b) Values per variable indicated per location

9. Save visualization file for future editing.

For most parameters needed by the map, heuristics are given by Dr. Kohonen's study [3].

D. Significance of the Study

Multivariate data that are obtained through different surveys have potential underlying patterns and relationships that can be used for researches. The implemented tool devises a way on visualizing data to obtain the potential relationships.

Patterns that are determined through the visualization tool provides possible hypotheses and relationships which can further be tested statistically.

Also, results that are obtained through the Philippine Regional Health Data used for this study allows for comparisons between the current health status of the different regions as well as assessing similarities between clusters in the data.

E. Scope and Limitations

The main purpose of this study is to produce a tool that would enable the visualization of correlations, and clusters of multivariate data. Limitations of the study include the following:

1. The tool does not in any way attempt to replace statistical softwares.

2. The tool shall only accept imported input data that are in .xls, or .csv format and those that comply to a specific data layout which include data labels and variable headers.

3. The U-matrix does not always give the optimal clusters of the data. Because the colors of the map are relative to how far the values range it may show a map that is dark. Deciding on the clusters is, however, subject to the user.

4. The tool uses the k-means algorithm. By nature, the k-means algorithm may give different clusters everytime it is run. It is suggested that the user runs the algorithm several times and choose whichever cluster seems best according to preference.

5. Visualization through GIS should contain data that are geographic in nature. The tool queries a database that contains the names of the different places as well as their bounds. If the database does not recognize a place from the input data. No visualization shall be shown for that location.

6. Visualization through GIS requires an internet connection. Google Maps requires the user to input their API key whenever the map is used. However, the tool generates an XML file that contains the data even without connection.

7. The tool uses the rectangular topology of the lattice.

8. It is not the tool's functionality to add the bounds of a specific place into the database.

9. Visualization includes the Unified Distance Matrix, Clusters, Component Planes, and GIS only.

10. The visualization tool would have its specific file type (.vis).

F. Assumptions

Because of the nature of data visualization as well as the self-organizing map. This study assumes the following:

1. The user has a background on the self-organizing map and its parameters.

2. Deciding on hypotheses may be subjective. Visualizations come together with the range of values. Similar to level of significance in statistics, the user may label the pattern significant or not.

3. Computers to be used have its monitors/projectors in good shape. Colors may change depending on the monitor's capacity.

II. Review of Related Literature

Kohonen has defined the self-organizing map to be an artificial neural network that uses unsupervised learning algorithm to adjust a map in parallel to input signal patterns. These maps are normally used for data visualization wherein pattern recognition and clustering are sought [3, 9].

In a research done by Basara and Yuan, the population health of 511 communities was visualized using self-organizing maps by considering 92 variables that come from physical, economic, occupation, housing, education and demographic environments. The data was successfully grouped into 5 clusters. In visualizing these data, The SOM determined clusters were reprojected to a geographic map and were compared with the distributions of different health outcomes. These health outcomes include Hepatitis A, Tuberculosis, Asthma, Fetal Immaturity, Diabetes, Influenza, Atherosclerosis, Alcohol and Drug, and Stress. Statistical analysis of variance was used to test for the significance of the community clusters in predicting the distribution of disease occurence. The study demonstrated a positive relationship between environmental factors and health outcomes and the SOM-GIS method was proven to be feasible in overcoming common methodological challenges that are encountered in research [10].

Koua and Kraak have also used the self-organizing map in visualizing demographic and health data from 152 countries using 30 basic indicators for health and living standard. These indicators include health expenditure, health risk factors, mortality, and reproductive health. The authors suggest that the use of SOM may improve geographical analysis of large and complex datasets since neural networks have the ability to handle noisy data in non-ideal contexts. In the exploration of general patterns and clustering, the unified distance matrix was primarily used to represent the data. However, alternative representations of the SOM were also presented including a geographical map that is more likely regarded as a Geographic Information System. On the other hand, the exploration of correlations and relationships within the input data was visualized through the use of component planes. These component planes show the values of the map elements for different attributes and how they vary over the space of the SOM units. The paper has successfully given an effective

application of the self-organizing map in the extraction of patterns and relationships in large data sets [11].

Apart from health data, the self-organizing map has been also used in visualizing complex geospatial datasets. In another study done by Koua, data including 29 socio-economic indicator variables such as population and habitat distributions, urbanization indicators, income of inhabitants, family and land data, industrial, commercial, and non-commercial data were considered to assess similarities, patterns, and relationships between municipalities in Overijssel region in the Netherlands. The study aimed to explore ways to support researchers in extracting information from large geospatial data sets. Very similar to the other work stated, Koua used a unified distance matrix to visualize clusters in the data set. Other visualizations such as mesh, component and surface plots were also used to visualize other information [12].

Another study that employed self-organizing map was done by Blazejewski and Coggins in a study about high-frequency financial data. The main objective of the study was to perform a qualitative analysis of the trade dataset. Input dataset consisted of 1,059,714 trades, each with 4 variables; trade size, best bid, ask volumes, and previous trade directions. However, to reduce the number of datasets, an aggregation of trades that were triggered by the same order was made. The study has successfully demonstrated the application of the self-organizing map to cluster financial trades into buyer-initiated and seller-initiated groups. The self-organizing map's capability to emphasize on the occuring patterns was also noted versus a simple histogram that failed to separate rare and common trading behaviors. The authors also suggested the inclusion of more variables for further studies [13].

The self-organizing map has also been used in visualizing parliamentary election results. In a study done by Niemela and Honkela, the results of parliamentary elections over a period of time were organized together with several political and socio-economic variables. The study aimed to study the effect of socio-economic conditions on the approval ratings of the different parties involved in the Finnish parliamentary elections from 1954 to 2003. A total of thirty-three variables were used in the study, eleven election variables, twelve economic condition variables, and ten government responsibility variables. The common belief that being in the government causes a popularity reduction for the next election was

proven by the study through visualizing the four largest parties involved in the government. Several other relationships such as a negative correlation between the approval ratings of the Centre Party versus inflation, and unemployment have also been seen. These relationships may have occurred because of the position of the Centre Party as the largest party in the government. The study has successfully explored the relationships between parliamentary election and socio-economic indicators and has suggested the use of a self-organizing map as a bridge between qualitative and quantitative methods. In particular, authors suggest the use of the specific findings seen using the self organizing map as hypotheses in further studies [14].

Though the self organizing map has been mostly used in looking for relationships in data, it may also be used for studies that would aid several people in decision making. Cottrell et al. used the self-organizing maps in visualizing successive multidimensional data measured on aircraft engines. Upon projecting the data on the self-organizing map, trajectores of the data were followed over time to determine deviations from normal behavior and thus anticipate failures. In the study however, the authors used the SOM together with a General Linear Model to eliminate possible effects of engines and environmental conditions to the data. Residuals that were taken from the linear model were used as inputs to the SOM. Each flight of an engine per variable were then plotted to obtain the trajectories. After such, the next step was to characterize the different shapes of the trajectories, define a suitable distance measure between the trajectories, and define typical behaviors related to typical faults. Hence, the SOM has proven its capability in anticipating faults in aircraft engines by the visual examination of such trajectories [15].

Lastly, Penn used the self-organizing map in the visualization of geochemical and hyperspectral data. The feasibility of use of the SOM for such is supported by the data's characteristics which are generally complex and highly-dimensional in nature and possibly depicting subtle relationships. For major element data, 11 variables were used as elements for the vectors while hyperspectral data used 2151 variables. The study used a one-dimensional SOM apart from the common 2-3 dimensional map. And the SOM has been proven to be a quick way to analyze major element datasets objectively [16].

The self-organizing map comes together with several visualization techniques for differ-

ent objectives. In some cases, the self-organizing map is used to arrange and cluster data while others involving geographically related data prefer to put the values onto a geographic map.

Vesanto and Alhoniemi conducted a study on clustering of the self-organizing map. In particular, the study performed clustering through two different approaches, the heirarchical agglomerative clustering and partitive clustering using k-means. Clustering was achieved through a two stage process, training through the self-organizing map and applying a clustering algorithm. The authors found out a significant difference between the times of clustering the data directly and clustering prototypes produced through the self organizing map [5].

Fincke et. al has performed a visualization of the self-organizing map through the use of a GIS. Because of its powerful algorithm, the self-organizing map can depict patterns in large datasets. In cases however that geographically-related data are involved, it is vital to perform spatial analysis to the data. Also, the study has devised a way to import data to a GIS software thus enabling spatial analysis to be performed [6].

The preceding articles have proven that the use of the self-organizing map for visualizing multidimensional data is feasible and particularly useful for data mining. Hence, the self-organizing map together with geographic information systems may be used to visualize world health data and perform initial processing for futher analysis.

III. Theoretical Framework

A. Data Visualization

Data Visualization is the study of the visual representation of data, meaning "information which has been abstracted in some schematic form, including attributes or variables for the units of information [2]." Information Technology combines the principles of visualization with powerful applications and large data sets to create sophisticated images and animations. An evident example is a tag cloud that uses text size to indicate relative frequency of use of a set of terms. Data that feed the tag cloud come from thousands of Web pages, representing millions of users. With visualization, information is contained in a simple image that everyone can understand quickly and easily.

A lot of instructors in the academic setting prefer to use data visualization to help their students understand concepts well. While researchers in a wide range of academic disciplines use visualizations to present data in ways that help generate new knowledge and understanding [17].

B. Self-Organizing Map

Self-organizing map (SOM), also known as Kohonen map, is a type of neural network based on competitive learning. It can be used for clustering and for visualization of high dimensional data by representing them in much lower dimensional spaces. It provides a topology preserving mapping from the high-dimensional space to map units. Map-units, or neurons, usually form a two-dimensional lattice and thus the mapping is from high-dimensional space onto a plane. The property of topology preserving means that the mapping preserves relative distance between the points. Points that are near each other in the input space are mapped to nearby map units in the map. Thus, the SOM can serve as a cluster analyzing tool of high-dimensional data [18].The Self-Organizing Map is a two dimensional array of neurons:

$$M = \{m_1, m_2, ..., m_{p \times q}\}$$

One neuron is a vector called the codebook vector:

$$m_i = [m_{i1}, m_{i2}, ..., m_{in}]$$

These neurons have the same dimension as the input vectors (n-dimensional) which contains the elements called weights. They are connected to adjacent neurons by a neighborhood relation. This dictates the topology, or the structure of the map. Usually, neurons are connected to each other either via rectangular or hexagonal topology. Figure 1 shows the rectangular topology on the left and the hexagonal topology on the right.

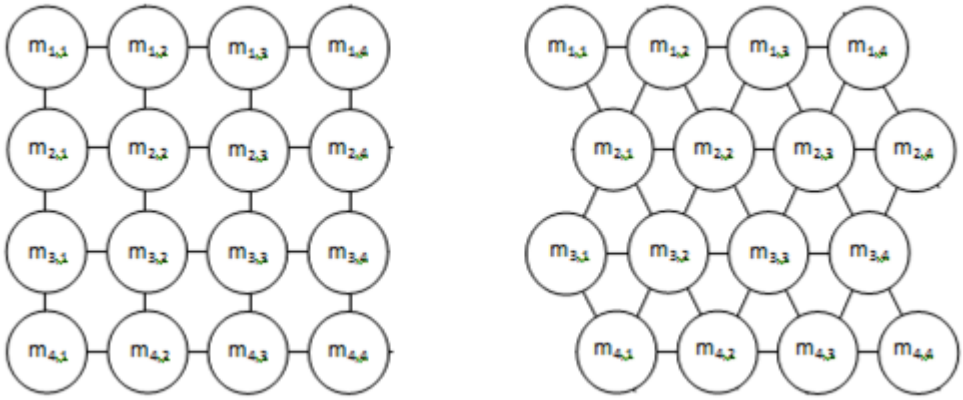

Figure 1: The Rectangular and Hexagonal Topologies

Distance between the map units may also be defined according to their topology relations. Immediate neighbors (adjacent neurons) belong to the neighborhood N_c of the neuron m_c. The neighborhood function is a decreasing function of time: $N_c = N_c(t)$. Like any other artificial neural network, the self-organizing map has a learning algorithm. However, a self-organizing map does not need a target output to be specified unlike many other types of network. Training a SOM occurs in several steps and iterations:

1. Each node's weights are initialized.

   Prior to training the data, each node in the lattice should be initialized, typically set to small standardized random values. The nodes may be randomly initialized or randomly take samples from the data and use it for initialization.

2. An input vector is chosen at random from the set of data and presented to the lattice.

3. Every node is examined to calculate which one's weights are most like the input vector. The winning node is known as the Best Matching Unit (BMU).

   To determine the best matching unit, we commonly iterate through all the nodes in the lattice and recommend to calculate the Euclidean distance between the each node's weight vector and the current input vector. The Euclidean distance is given by:

$$Dist = \sqrt{\sum_{i=0}^{n} (V_i - W_i)^2} \tag{1}$$

   where V_i is an element of the current input vector and W_i is an element of the node's weight vector. However, in the tool, other distance measures that the user prefers may be used.

4. Determine the BMU's local neighborhood. The radius of the neighborhood is calculated through the neighborhood function. Generally, this is a value that starts large, typically set to the radius of the lattice, but diminishes each time-step. Any node found within this radius are said to be inside the BMU's neighborhood.

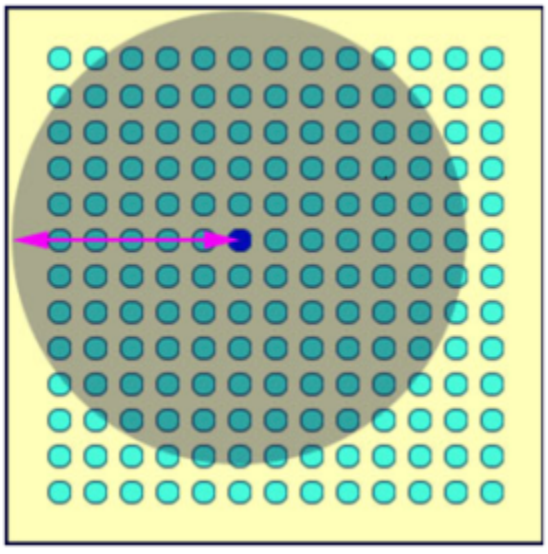

Figure 2: The Best Matching Unit and Its Neighborhood

In the figure above, the neighborhood is centered on the BMU and contains the most of the other nodes and the arrow shows the radius of the neighborhood. A unique feature of the Kohonen learning algorithm is that the area of the neighborhood shrinks over time that it is accomplished by reducing the radius over time. The default neighborhood function for the tool is given by:

$$\sigma(t) = \sigma_0 \exp\left(-\frac{t}{\lambda}\right) \qquad t = 1, 2, 3, ... \tag{2}$$

Where σ_0 denotes the width of the lattice at time t_0 and λ denotes a time constant which is computed based on the number of iterations set and the current size of the neighborhood; t is the current time step which in implementation is the current iteration of the loop. The tool allows for revisions in the neighborhood function but generally it should shrink over time. The figure below shows how the neighborhood shrinks over time assuming the the neighborhood remains centered on the same node. Generally, the BMU will move around depending on the current input vector presented to the network. Over time the neigborhood shrinks to the size of one node.

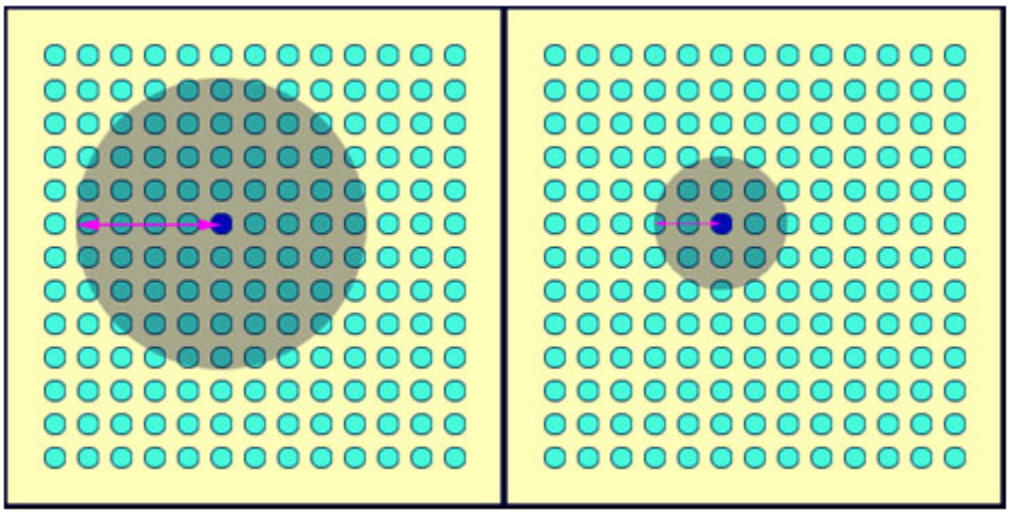

Figure 3: Neighborhood shrinks over time

5. Each neighboring node's weights are adjusted to make them more like the input vector. The closer a node is to the best matching unit; the more its weights get altered.

Basically, to determine whether a node is within the neighborhood, we iterate through all the nodes to determine if they lay within the radius or not. Every node within the BMU's neighborhood has its weight vector adjusted according to the following equation:

$$W(t+1) = W(t) + \Theta(t) L(t) (V(t) - W(t)) \tag{3}$$

Where t represents the current time step, L is a variable called learning rate which decreases with time, and Θ is the amount of influence a node's distance from the BMU is on its learning. The equation states that the new adjusted weight for the node is equal to its current weight W, plus a fraction of the difference $\Theta \times L$ between the current weight and the input vector. This function is given by Dr. Kohonen in his paper and is universally accepted by pioneers in the self organizing map [3].

The preset decay of the learning rate is computed using the following equation:

$$L(t) = L_0 \exp\left(-\frac{t}{\lambda}\right) \qquad t = 1, 2, 3, ... \tag{4}$$

This is basically similar to the neighborhood function but is used to decay the learning rate. However, the tool allows the users to set this function according to their preference.

The tool also allows for other learning decay functions including the power series:

$$L(t) = L_0 \left(\frac{L_f}{L_0}\right)^{t/T} \qquad t = 1, 2, 3, ... \tag{5}$$

Where L_f is the final learning rate and T is the total number of iterations. On the other hand, Θ by default is computed using:

$$\Theta(t) = \exp\left(-\frac{dist^2}{2\sigma^2(t)}\right) \qquad t = 1, 2, 3, ... \tag{6}$$

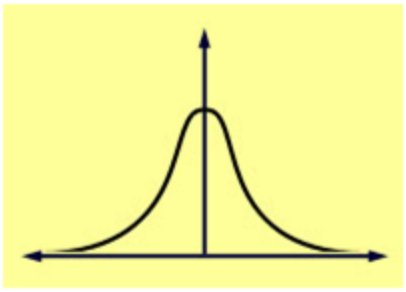

Figure 4: The Gaussian Decay function

Where *dist* is the spatial distance of the node from the BMU and σ is the width of the neighborhood function as calculated using equation (2).

6. Steps 2-5 are repeated for N iterations.

Though referred to as a neural network, self-organizing maps work differently from most known neural network algorithms which are commonly supervised. With regard to the SOM's convergence criterion, the SOM limits its training algorithm to a specific number of iterations. Since SOM is an unsupervised learning technique, there is no target vector and thus, we cannot obtain a difference between a target vector and the value at the current iteration [19]. And since the learning is a stochastic process, the statistical accuracy of the mapping depends on the number of iterations which must be reasonably large, Kohonen has devised a "rule of thumb" for the number of iterations in training[3].

C. Unified Distance Matrix

The Unified Distance Matrix (U-Matrix) is a visualization technique that is constructed on top of a self-organizing map [20]. Let n be a node in the map, $N(n)$ be the set of immediate neighbors on the map, and $w(n)$ be the associated weight vector for node n, then:

$$U - height\,(n) = \sum_{m \in N(n)} d\,(w\,(n)\,, w\,(m)) \tag{7}$$

where $d\,(x, y)$ is the distance used in the SOM to construct the map. The U-Matrix generally displays the U-heights on top of the grid positions of the neurons on the map. U-matrices are usually displayed as a grey level picture. It has become a standard tool

for displaying distance structures of the input data on the SOM. The U-matrix has several properties:

1. The position of the projections of the input data points reflect the topology of the input space inherited from the underlying SOM algorithm.
2. Weight vectors of neurons with large U-heights are very distant from other surrounding vectors in the data space.
3. Weight vectors of neurons with small U-heights are very near from other surrounding vectors in the data space.
4. Dark colored nodes depict cluster boundaries.
5. Light colored nodes depict a cluster.

Basically, the U-matrix visualizes distances between nodes in the self-organizing map. This technique enables a visualization of clusters in data and particularly good for finding the number of clusters in datasets where it is not predetermined.

D. K-means clustering algorithm

The k-means clustering is an algorithm to group objects based on its attributes into k number of groups. This is done by minimizing the sum of squares of the distances between the data and the corresponding cluster centroid [21].

The k-means algorithm is a step by step process. Here is a breakdown of the process:

1. Decide on the value of k, the number of clusters.
2. Put an initial partition that classifies the data into k clusters. The initial partition may be randomly assigned. However a systematic process may also be made by using the following:
   (a) Take the first k samples as single-element clusters.
   (b) Then, assign each of the remaining samples to the cluster with the nearest centroid and recompute the centroid of the gaining cluster after each assignment.

3. Take each sample in sequence and compute its distance from each of the centroids. Then assign the sample point to the nearest cluster by minimizing the distance of the point to the centroid.

4. Check for changes in clusters. If there is, then repeat the third step. If none, then terminate the loop and the final clusters have been reached.

For a certain case that the number of clusters given is greater than the number of data(nodes), then we assign each data point a centroid and thus, each node is considered to be a cluster.

We continuously adjust the locations of the centroids by recomputing through the average of the values included in a certain cluster the since we are not sure about their true positions. Mathematically, this iteration is proven to be convergent. Since there is only a finite set of possible clusterings, the algorithm will eventually arrive at a minimum.

E. Geographic Information Systems

A Geographic Information System is a system designed for storing, analyzing, and displaying spatial data [22]. These manipulations are primarily accomplished through computers with the help of several software. Through GIS, we are able to carry out spatial analysis for data that are geographic in nature.

Spatial Analysis is simply an analysis of spatial data and gives information that are spatial in nature. Saying that the data is spatial in nature, the answer involves geographic areas. To present these answers we use themes.

A theme is a way of the GIS to present different features that exist in the geographic space. The GIS stores these features as separate data layers. Themes are stored in different types. Points are used for locations and landmarks. Lines are used for roads and other similar structures. And polygons are used to depict a certain area. For thematic maps, we use polygons with colors to show values and distinguish a certain area from another.

IV. Design and Implementation

A. System Design

The SOM Visualization Tool's Context Diagram is shown on Figure 5. The figure suggests that the tool should be able to render different visualizations of the data through the use of the self-organizing map by asking the user several inputs including the dataset, and details about the map and the algorithm. However, GIS visualization would only be applicable for data that are geographically related. Also, the tool provides a tutorial for the usage of the tool.

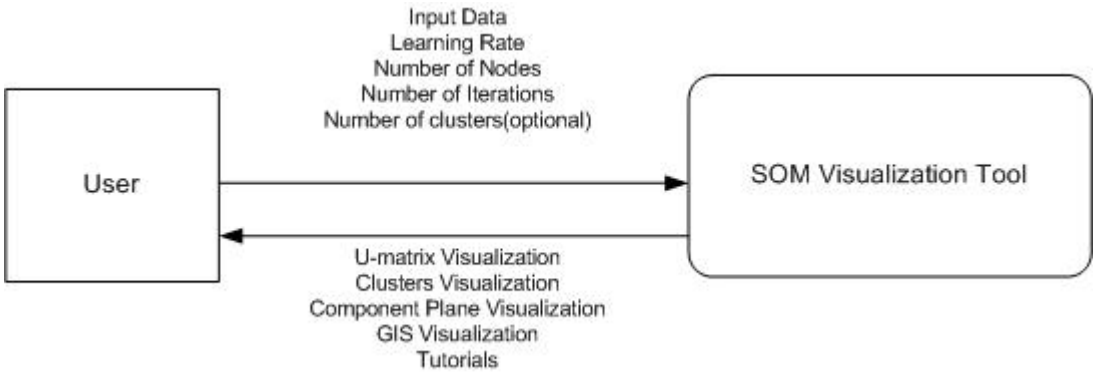

Figure 5: Context Diagram, SOM Visualize

The tool will be implemented using an object-oriented model. Its graphical user interface shall be divided into three major parts: the input panel, the console panel, and the results panel.

The **Input panel** shall include all fields that would be needed for visualization. The input panel shall consist of sub-panels including the SOM panel, U-matrix panel, Clusters panel, and GIS panel. Each sub-panel would consist of inputs that are specifically needed for their algorithms.

The **Console panel** shall be responsible of logging events that are triggered by the user as well as errors encountered.

The **Results panel** shall include all the visualizations that were generated by the tool. The results panel shall be displaying one visualization at a time depending on what the user wants to see.

The use-case diagram of the SOM Visualization tool is shown in Figure 6.

Figure 6: Use Case Diagram, SOM Visualize

The use case diagram shows the different functionalities that the user can do with the tool. In order to completely access all functionalities, the user must start by creating a file. The tool shall be asking for the input data and validate it. Other map details shall be asked once the input data has been successfully validated. Once done, the self-organizing map will be trained and will enable the viewing of the different visualizations generated by

the tool. In GIS visualizations, geographic locations will be checked for its existence in the geographic map. The user may also choose to view the tutorial to learn more about the tool and how to use it.

The top level data flow diagram is shown on Figure 7.

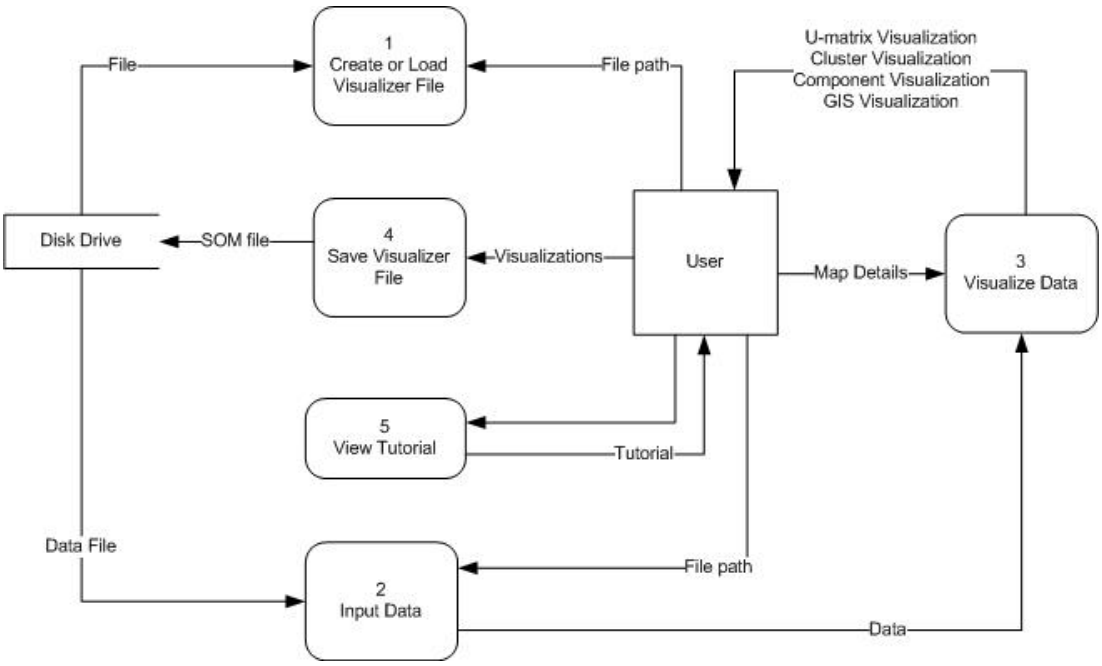

Figure 7: Top Level Data Flow Diagram, SOM Visualize

The SOM Visualization Tool has five main processes; creating/loading a file, inputting the data, visualizing the data, saving the visualization file, and viewing the tutorial. Loading a file is a process wherein saved works in the visualization tool are opened for further editing. The second process, inputting the data, is basically a process that enables the user to supply data for the tool. This process includes both an export facility, for data that are in excel format or manually inputting the data through the tool. The last process, saving the visualization file, is a process where works that are needed for further use are stored into the hard disk. Among all of these, the third process, visualizing the data, incorporates most of the functionalities of the tool and outputs the different visualizations that are to be generated by the system. The sub-explosion of this process is shown on Figure 8.

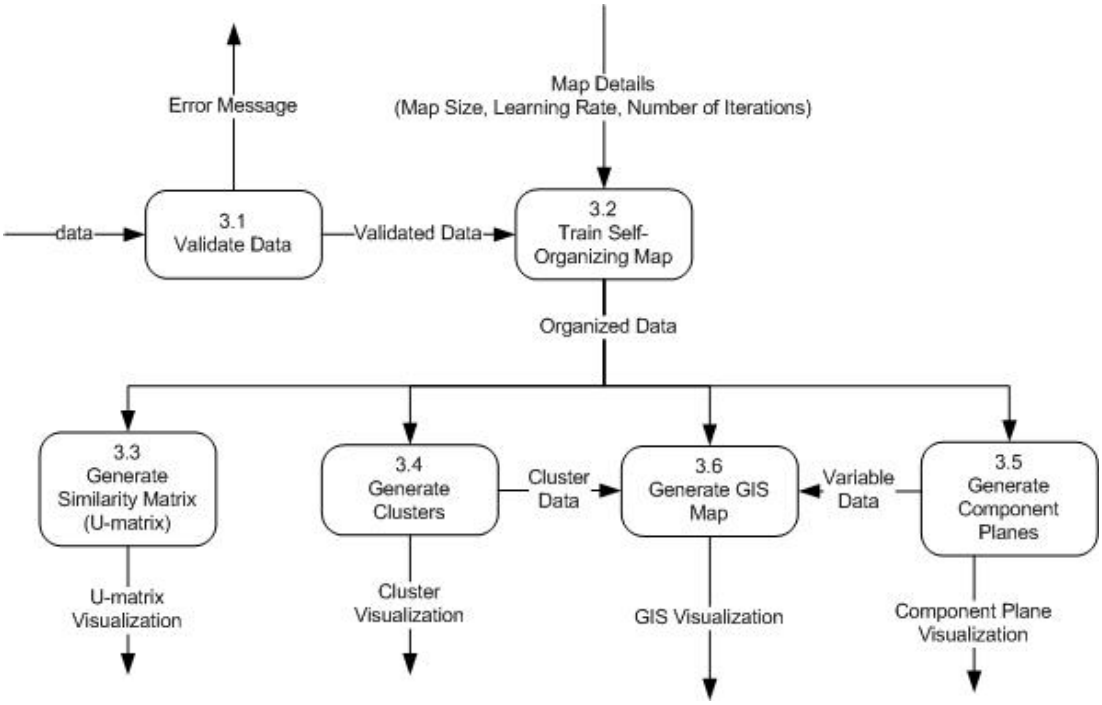

Figure 8: Sub-explosion of Process 3: Visualizing the Data, SOM Visualize

Data supplied by the user is validated by the system, upon validation and supplying the map details, the map is trained for organization of data. From this map, visualizations of different types may be made.

The process for training the map is detailed on Figure 9. The map is initialized by randomizing weights for all the nodes. Iteratively, an input vector is chosen at random and presented to the map. The Best Matching Unit is computed by measuring the minimum distance according to the Euclidean formula as seen in Figure 10. After such, the radius of the neighborhood and the current learning rate is determined through the use of the time constant, the map size, and the initial learning rate. Upon computing the two, the neighborhood's weights are adjusted according to a formula that uses the current learning rate and neighborhood radius as seen on Figure 11.

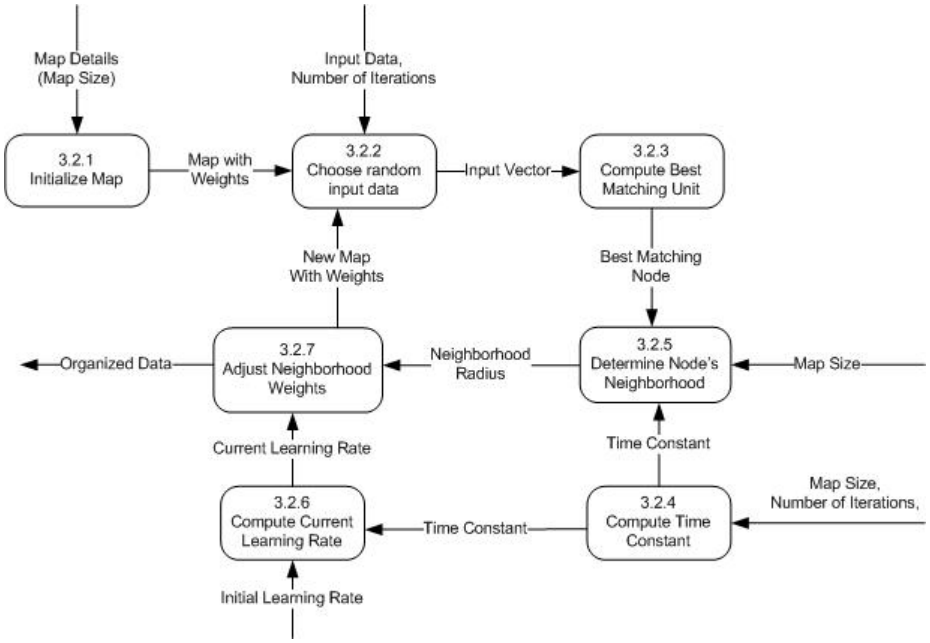

Figure 9: Sub-explosion of Process 3.2: Training the Map, SOM Visualize

Computing the Best Matching Unit is an iterative process. We iterate through all the nodes in the map, and compute the Euclidean distance between the chosen node and the input vector. We compare with the minimum distance and return whichever is less and store the node and distance. Upon completion, we return the Best Matching Unit.

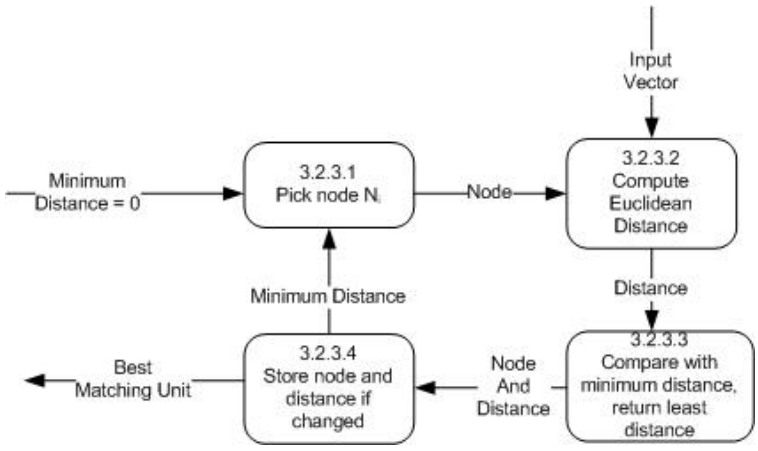

Figure 10: Sub-explosion of Process 3.2.3: Computing the Best Matching Unit, SOM Visualize

Adjusting the neighborhood weights is also an iterative process. We iterate through all nodes that are considered neighbors of the Best Matching Unit and compute their distances.

Upon knowing the distance and the neighborhood radius the BMU's influence to the node is computed followed by the adjustment of the weights.

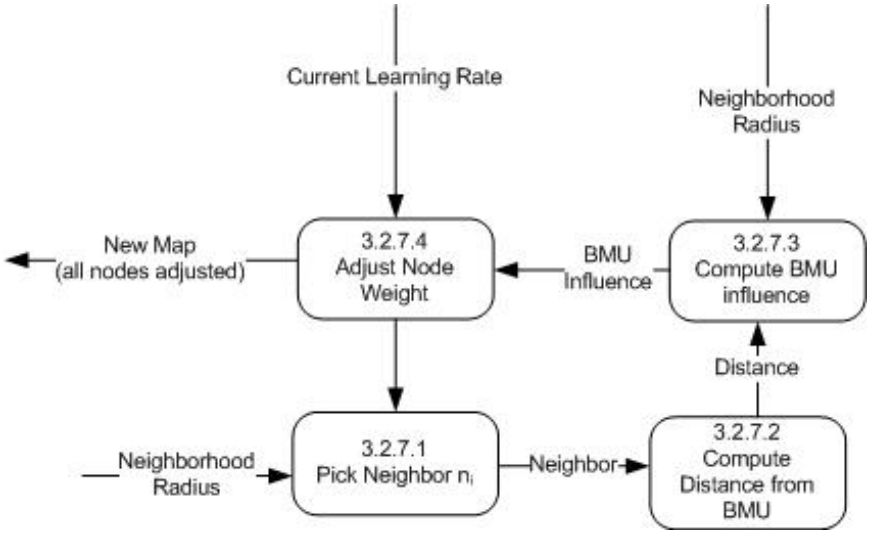

Figure 11: Sub-explosion of Figure 3.2.7: Adjusting the Neighborhood Weights, SOM Visualize

The following algorithm summarizes the training of the self-organizing map:

```
1.) Initialize the map by iterating through
all nodes and using this function

public Node(int left, int right, int top, int bottom, int NumWeights)
  {
    //initialize the node weights to small random variables
    for w in 0:NumWeights
    {
      nodeWeights.push(RandFloat());
    }

    //calculate the node's center
    m_dX = m_iLeft + (double)(m_iRight - m_iLeft)/2;
    m_dY = m_iTop  + (double)(m_iBottom - m_iTop)/2;
  }

2.) Get the Distance by iterating through all the nodes
to determine Best Matching Unit

for size in 0:SOM.size()
{
distance = currNode.getDistance(inputVector);
if(size = 0){
```

```
    minDistance = distance;
}
if(distance < minDistance)
{
minDistance  = distance;
BMU = currNode;
}

return BMU;
}
double Node->getDistance(const vector<double> &InputVector)
{
  double distance = 0;

  for i in 0:nodeWeights.size()
  {
    distance += Math.squared((InputVector[i] - m_dWeights[i]));
  }

  return sqrt(distance);
}
```

3.)The entirety of the training algorithm is as follows:

```
bool SOM->trainMap(const vector<vector<double> > &data)
{
  //check weights sizes
  if (data[0].size() != SIZE_OF_INPUT_VECTIRS) return false;

  //return if the training is complete
  if (m_bDone) return true;

  //enter the loop
  while (iterationCount < NUM_OF_ITERATIONS)
  {
    //randomize an input vector
    int currVector = RandInt(0, data.size()-1);

    //present the vector to each node and determine the BMU
    winningNode = FindBestMatchingNode(data[ThisVector]);

    //calculate the width of the neighbourhood for this timestep
    m_dNeighbourhoodRadius = m_dMapRadius *
            exp(-(double)iterationCount/m_dTimeConstant);
   //adjust weights

   //For each node calculate the influence
   for (int n=0; n<SOM.size(); ++n)
```

```
   {
     //calculate the Euclidean distance (squared)
     to this node from the BMU

     double distanceToNodeSquared =
          Math.squared((winningNode->X()-SOM[n].X())) +
      Math.squared((winningNode->Y()-SOM[n].Y()));
     double neighborhoodSquared = Math.squared(neighbourhoodRadius);

     //if within the neighbourhood adjust its weights
     if (DistToNodeSq < neighborhoodSquared)
     {
       //calculate by how much its weights are adjusted
       bmuInfluence = exp(-(distanceToNodeSquared)/
        (2*neighborhoodSquared));

       SOM[n].adjustWeights(data[ThisVector],
                            learningRate,
                            bmuInfluence);
     }

   }//next node

   //reduce the learning rate
   learningRate = INITIAL_LEARNING_RATE *
            exp(-(double)iterationCount/TIME_CONSTANT);

   iterationCount++;
  }

  else
  {
    m_bDone = true;
  }

  return true;
}
```

After applying the algorithm, an organized data is returned. This data shall then be subjected to visualization. The following diagrams are sub-explosions of the visualization techniques.

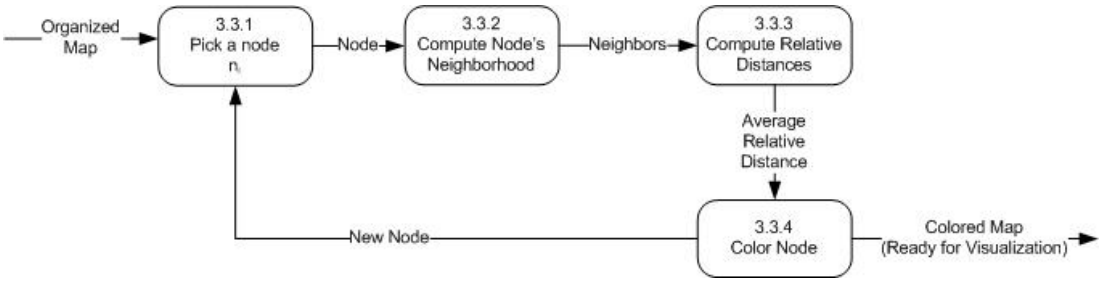

Figure 12: Sub-explosion of Process 3.3: U-matrix visualization, SOM Visualize

Figure 12 shows the process of visualizing the data using a distance matrix, specifically, the Unified Distance Matrix. The figure suggests that we iterate through each node and compute their average relative distances from their neighbors and color the node appropriately based on the value obtained.

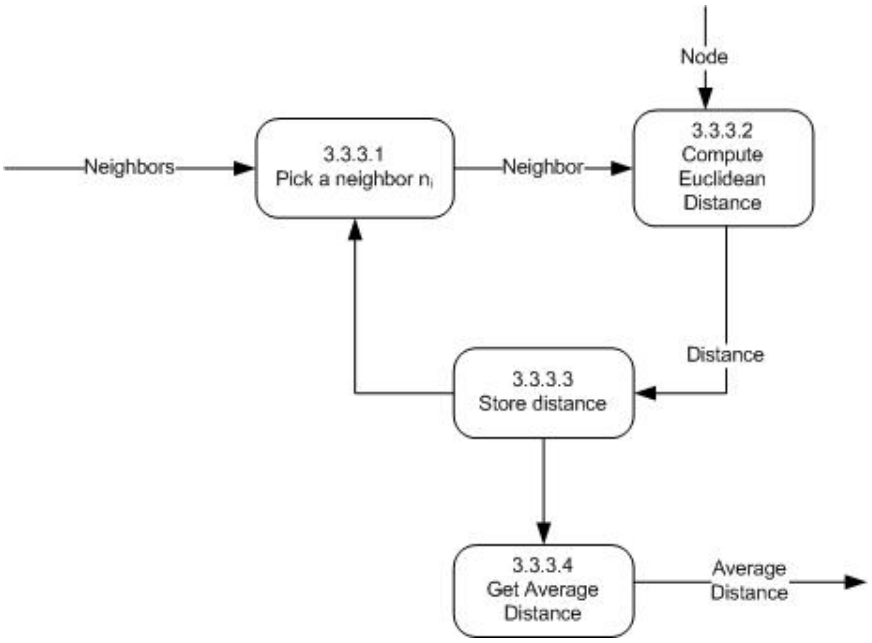

Figure 13: Sub-explosion of Process 3.3.3: Computing the Relative Distances, SOM Visualize

Figure 13 suggests that the average distance is taken by iterating through all the neighbors of a specific node.

The following pseudocode will be used to implement U-matrix visualization.

```
for n in 1:SOM.size()
{
neighbors = 0;
```

```
distance = 0;
currNode = SOM[n];

for i in 1:SOM.size(){
    distFromNodeSq =
        Math.squared(currNode.getX()-SOM[i].getX())*currNode.width()) +
        Math.squared(currNode.getY()-SOM[i].getY())*currNode.height());

    if(distFromNodeSq <= MATRIX_NEIGHBORHOOD_RADIUS_SQUARED){
        distance += currNode.getDistance(SOM[i]);
        neighbors++
    }


    averageDist = distance/(neighbors - 1);
    currNode.setColor(new Color(averageDist));
}
```

Figure 14 shows the flow of data for k-means clustering algorithm.

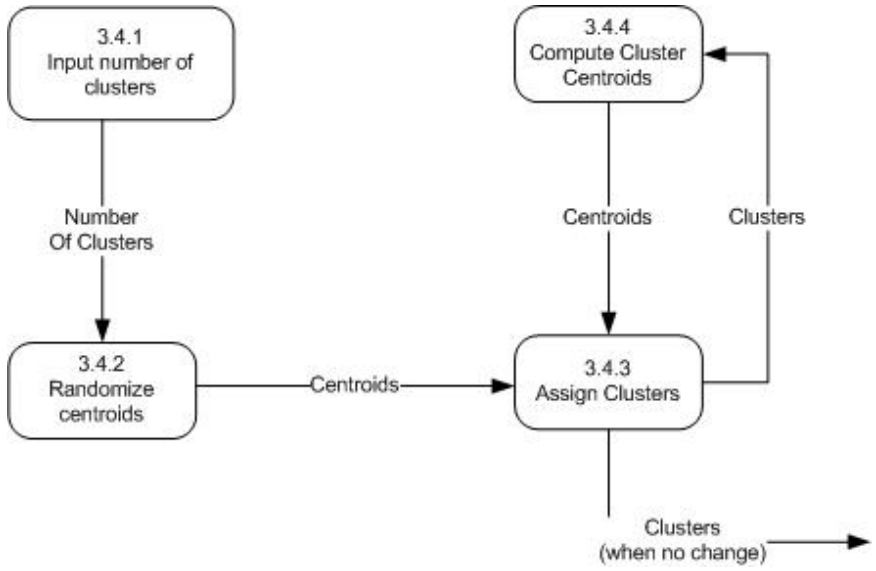

Figure 14: Sub-explosion of Process 3.4: Cluster Visualization, SOM Visualize

To properly cluster the values, the user must specify k - the number of clusters. Generally, the similarity SOM should guide the user on how many clusters he should input. After which, the centroids will be randomized for the first iteration and the assignment of clusters will be carried out. For optimization purposes, the clustering will be an iterative process until minimal to no change in clusters is encountered. Figure 15 shows how clusters

are assigned.

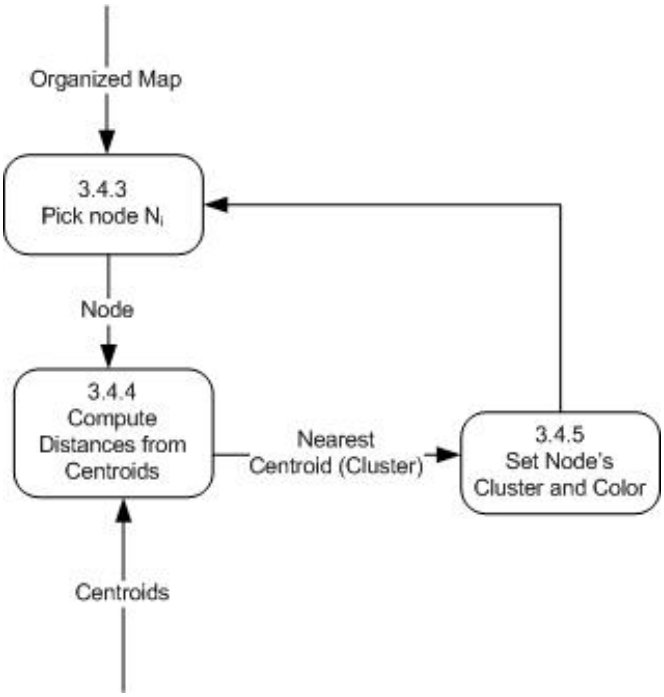

Figure 15: Sub-explosion of Process 3.4.3: Assigning Clusters, SOM Visualize

The following is a pseudocode of k-means algorithm:

```
for i in 1:k
{
    centroid[i] = centroid[i].push(randomWeights());

}

while(change){
    change = false;
    for j in 1:SOM.size()
    {
        for i in 1:k{
            distance = SOM[j].getDistance(centroid[i]);
            if(j == 1)
            {
                minDistance = distance;
            }
            if(distance < minDistance){
                SOM[j].setCluster(i);
                change = true;
            }
        }
    }
```

```
    for i in 1:k{
     centroid[i].weights = centroid[i].getNewCenters();
    }
}
```

Figure 16 shows the data flow for visualizing component planes. And Figure 17 shows how to color the map based on the variables.

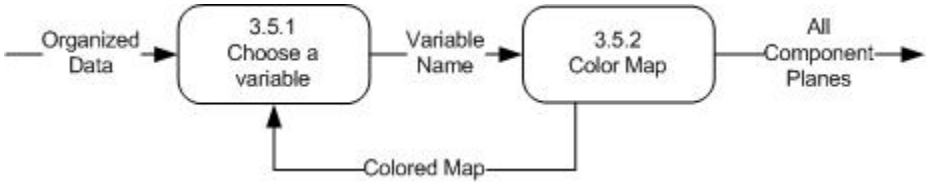

Figure 16: Sub-explosion of Process 3.5: Component Plane Visualization, SOM Visualize

Component Plane Visualization is simple. Per variable, the map is colored according to the values of each node with reference to the said variable.

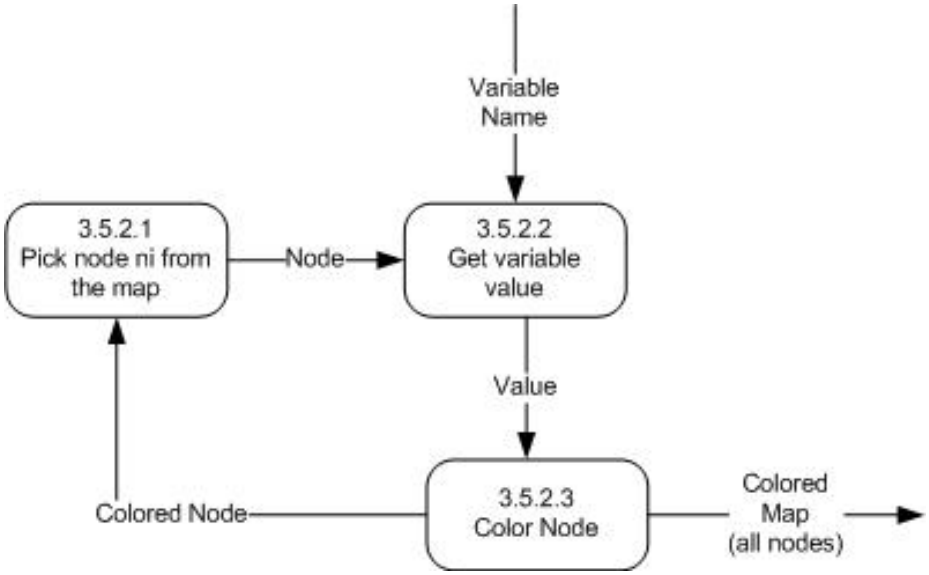

Figure 17: Sub-explosion of Process 3.5.2: Component Plane Coloring, SOM Visualize

The last figure, Figure 18 shows the process of GIS visualization. The concept is also pretty simple. The user may choose what values would he like to see on the map, a specific variable or clusters. To do such, data from the clustering algorithm and the component planes shall be used.

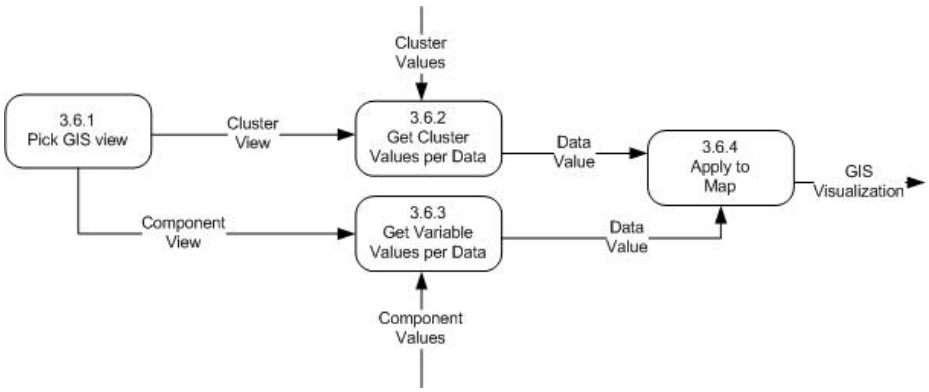

Figure 18: Sub-explosion of Process 3.6: GIS Visualization, SOM Visualize

B. Technical Architecture

Since the SOM Visualization Tool is implemented using Java, the tool will run on most operating systems provided that the system has Java Runtime Environment installed. However, it is recommended that the JRE is constantly updated to ensure compatibility.

Since the application also uses computationally expensive algorithms, it is also recommended to work on a computer with at least 1GB of RAM. This is to ensure that the application wouldn't run out of memory while doing its processes as well as in drawing graphics. Also, the application is best viewed using high resolutions (1024×768 or higher).

V. Results

SOM Visualize's main screen and welcome screen shows upon opening the application. In order to start using the application, the user should create a new file by clicking on the "New File" icon on the toolbar or by using the File Menu.

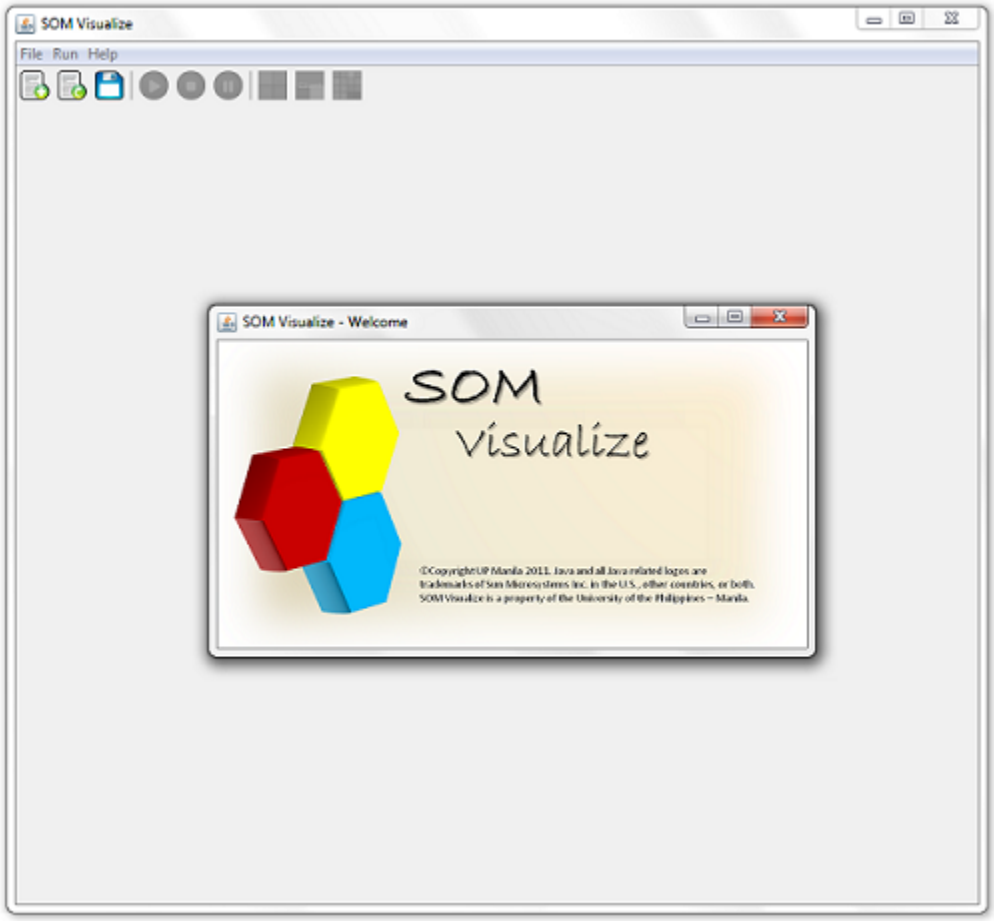

Figure 19: Main Window and Welcome Screen, SOM Visualize

Upon creating a new file, the application displays its work environment and the user may start using the application. SOM Visualize's work environment consists of three main panels: the Input Panel, the Console Panel, and the Result Panel. Figure 20 shows the application's work environment.

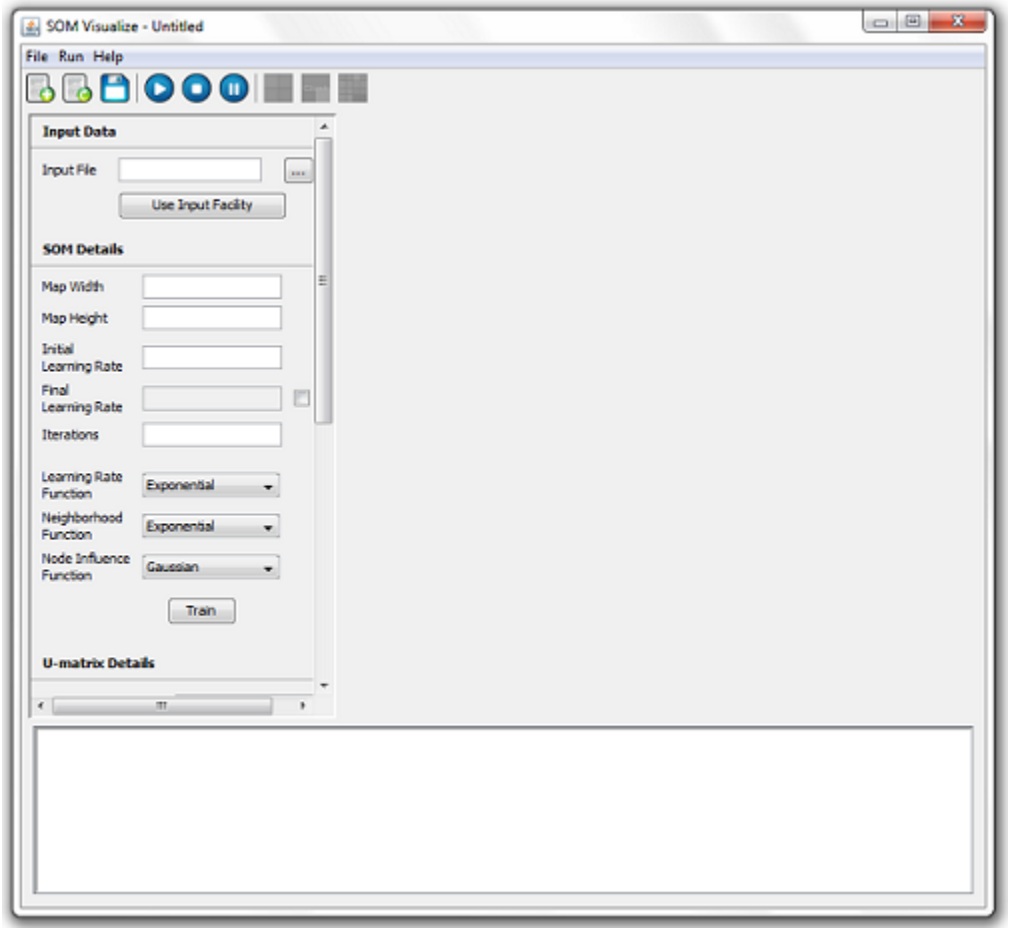

Figure 20: The Work Environment, SOM Visualize

Using SOM Visualize is an easy task. To start, the user should give the application all inputs that are asked for. All input boxes are contained within the Input Panel. This includes the File Input, SOM Parameters, U-matrix Visualization Parameter, Cluster Visualization Parameter, Component Planes, and GIS Visualizer. Figure 21 shows the content of the input panel.

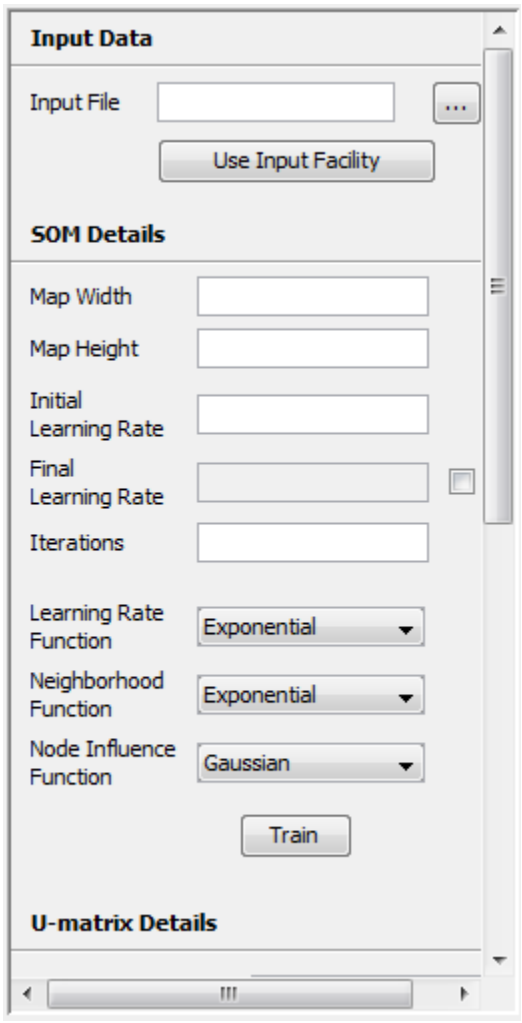

Figure 21: The Input Panel, SOM Visualize

The first thing to do is give the application the data to analyze. There are two ways to do so, the first way is to import an Excel File (.xls/.csv) to the tool, and the second way is to use the Input Facility Provided by the tool. The tool prioritizes the existence of an input file. This means that whenever text is given in the Input File Text Field, the tool shall try to locate for the path and the file. In order to use the input facility provided by the tool, the user should ensure that no input is given on the text field.

To import an excel file to the tool, the user may either choose to manually write the path of the file in the text field provided or use the browse button beside the text field and conveniently look for the file by using the file chooser. The tool only accepts two file extensions, an Excel 97-2003 format (.xls) or a CSV file. In the event that the file type is

invalid or the path cannot be recognized, the user shall be informed through the console. Figure 22 shows a screenshot of the valid layout for the excel files. Files imported to the tool should follow a specific data layout, which is, using column A of Rows 2-N as titles for input instances and using row 1 of columns B onwards for the variable titles. When erroneous data is encountered by the tool, training may proceed incorrectly or may not proceed at all depending on the error.

A	B	C	D	E	F	G	H	I	J	K
	Total Ferti	Median ag	Median ag	Median ag	Median wo	Women ag	Women cu	Women cu	Women cu	Women cu
NCR	2.3	23.7	22.6	24.8	62	18	54	32	9	14
CAR	3.3	21	20.7	22.1	61	26	55	39	15	13
Region I	3.4	22.7	22.3	23.8	61	29	54	36	11	18
Region II	4.1	21.1	20.9	22.3	67	32	54	46	8	27
Region III	3	22.4	21.9	23.5	62	26	58	40	17	16
Region IVA	3	22.9	22.5	24	61	27	47	32	10	14
Region IVB	4.3	20.3	20	21.6	63	37	54	36	6	22
Region V	4.1	21.4	21.3	22.6	70	27	39	24	7	11
Region VI	3.3	22.7	22.1	23.7	71	25	52	33	7	19
Region VII	3.2	21.9	20.8	22.9	67	26	56	36	7	14
Region VII	4.3	21.7	21.2	22.6	61	31	48	28	8	15
Region IX	3.8	21.7	20.9	22.8	58	31	44	29	4	19
Region X	3.3	21.6	20.8	22.7	64	27	53	39	6	18
Region XI	3.3	21.2	20.5	22.1	64	29	60	45	10	21
Region XII	3.6	20.6	20	21.5	65	35	55	41	12	19
Region XIII	4.3	21	20.5	22.3	67	34	52	37	9	15
ARMM	4.3	19.8	19.9	21.5	37	27	15	10	3	3

Figure 22: A valid imported data file, SOM Visualize

On the other hand, when using the input facility, the user should directly input the data following a specific data layout. The first column is alloted for the titles of each input instance and the next columns thereafter should contain the data. Figure 23 shows the appearance of the Input Facility provided by the tool.

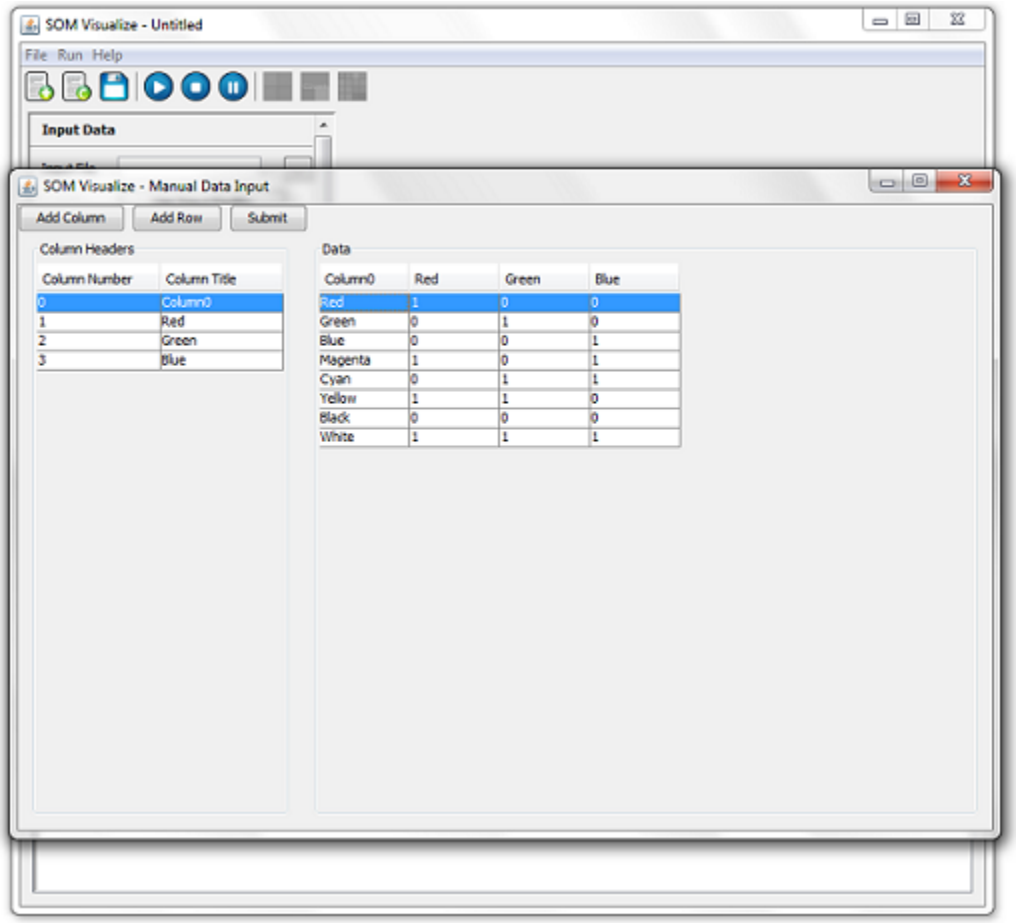

Figure 23: Using the tool's input facility, SOM Visualize

Data entered should always be numeric in nature except for data and variable titles. As in statistics, categories should also be represented by numbers.

Having the data entered, the next thing that the user should do is to give the Self-Organizing Maps's parameters. Parameters should include the Map Width, Map Height, Initial Learning Rate, Number of Iterations, Learning Rate Decay Function, Neighborhood Decay Function, and Node Influence Function. In order to guide the user, the tool provides a color coding scheme to indicate possible error in inputs. Figure 24 shows a set of possible input parameters for the map.

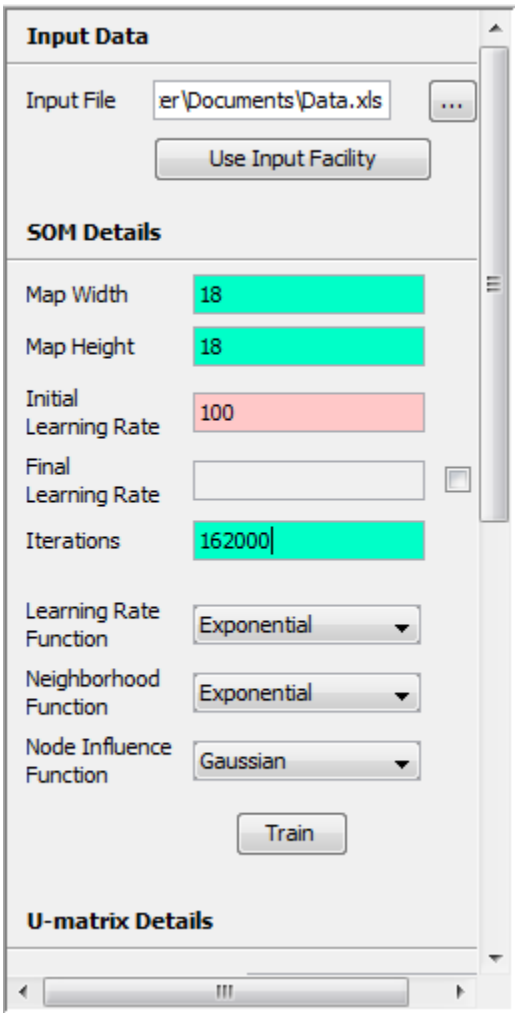

Figure 24: The Self-Organizing Map's Input Parameters, SOM Visualize

In the figure, all fields are colored green except for the Initial Learning Rate. Values for this field should not go beyond the range 0 to 1. In this case, the user's input is 100 and the tool marked the box red to indicate an error in input. Also, entering the Final Learning Rate is not necessary. The tool has a default final learning rate of 0. In cases where the learning rate decay function is set to the Power Series formula, the user may opt to indicate the learning rate at which the last iteration should run on.

Upon having all the correct inputs, the tool is ready to train its map. Training length varies depending on several factors. These factors include the size of the map, the input data dimension and count, the number of iterations, and other factors like the computer's processor speed, and memory available. The tool uses a computationally expensive algorithm.

Sometimes, it is necessary to leave the computer for several hours to allow for the map to successfully complete organizing the data. The tool's training algorithm is implemented using threads, this means that the user may opt to terminate the current training or pause the training and save the file to continue training later on. However, for a visualization to be seen, it is always needed that the map finishes the training step.

After successfully training the map, the user should completely see the work environment including the result panel. Figure 25 shows the appearance of the map after successfully training the data. After training, all visualization buttons seen in the input panel which includes the U-matrix, Clusters, and Component Planes shall be activated. The user then chooses which visualization to look at. However, the GIS Visualization Button will only be enabled when Component Plane Visualization and Cluster Visualization have been done.

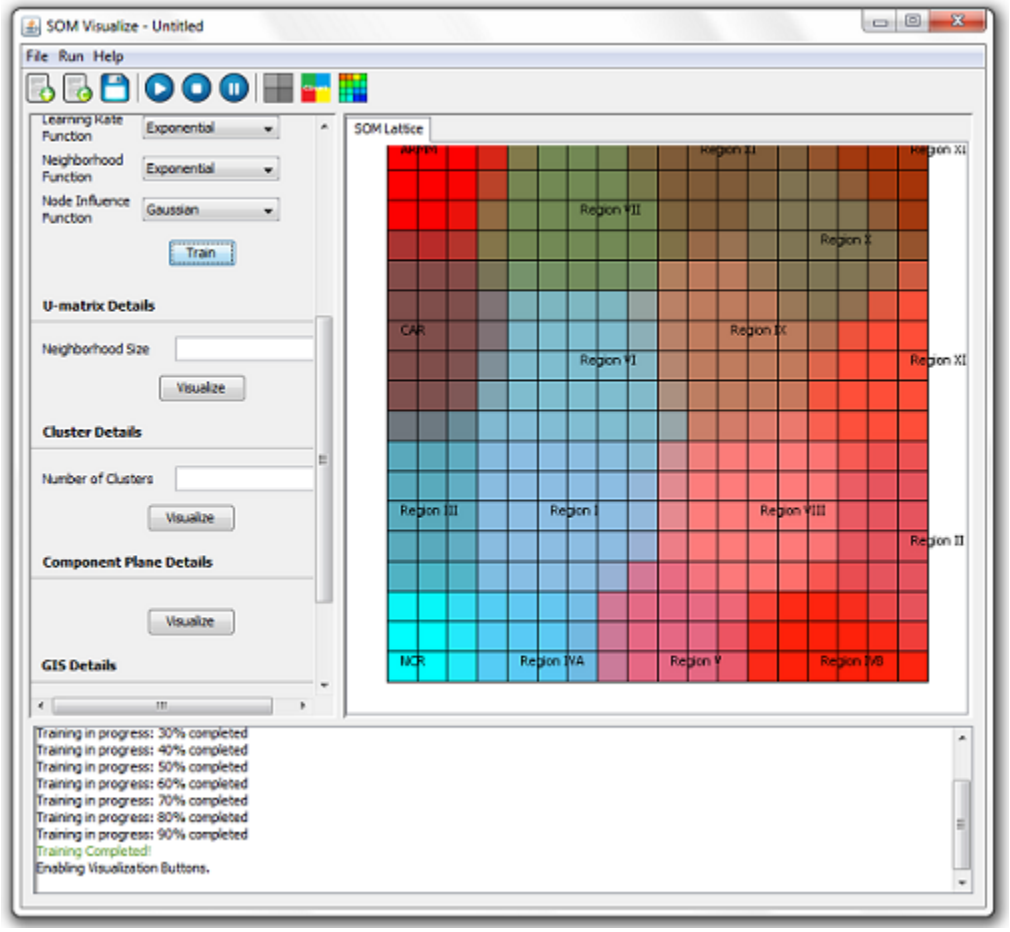

Figure 25: The Work Environment after successfully training the map, SOM Visualize

The result panel shall show whichever visualization is chosen by the user. It is a tabbed

panel where the different visualizations are placed. Figure 26 shows the application with the result panel containing four different visualizations: the initial lattice, the U-matrix, the clusters, and the component planes.

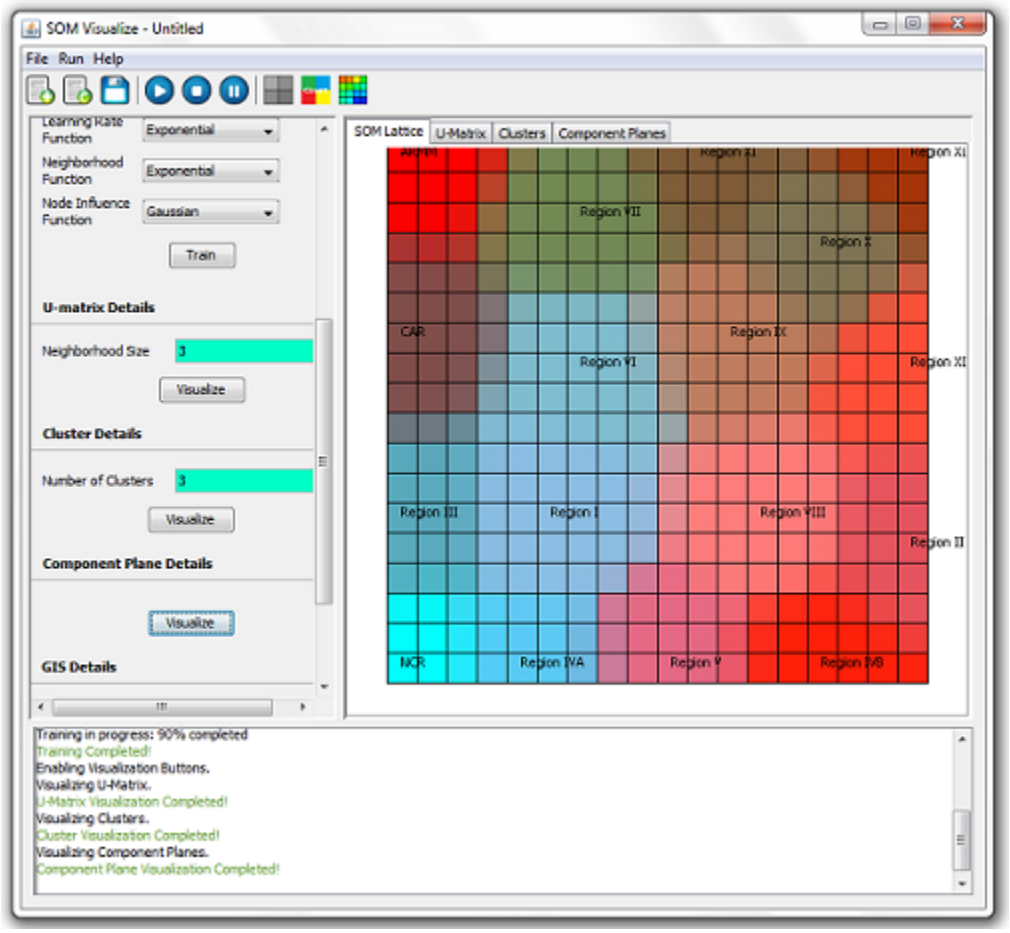

Figure 26: The result panel containing all four visualizations, SOM Visualize

Each tab of the result panel contains a different visualization. First, the initial lattice. The purpose of this tab is solely to inform the user of the positions of the dataset in the lattice. From the name itself, the initial SOM Lattice is present to give the user an initial impression on how the data was organized. On the event that the user is not satisfied with the appearance, retraining the map may be an option. Figure 27 shows the initial lattice with the Philippine Regional Health Data that has undergone a fast training.

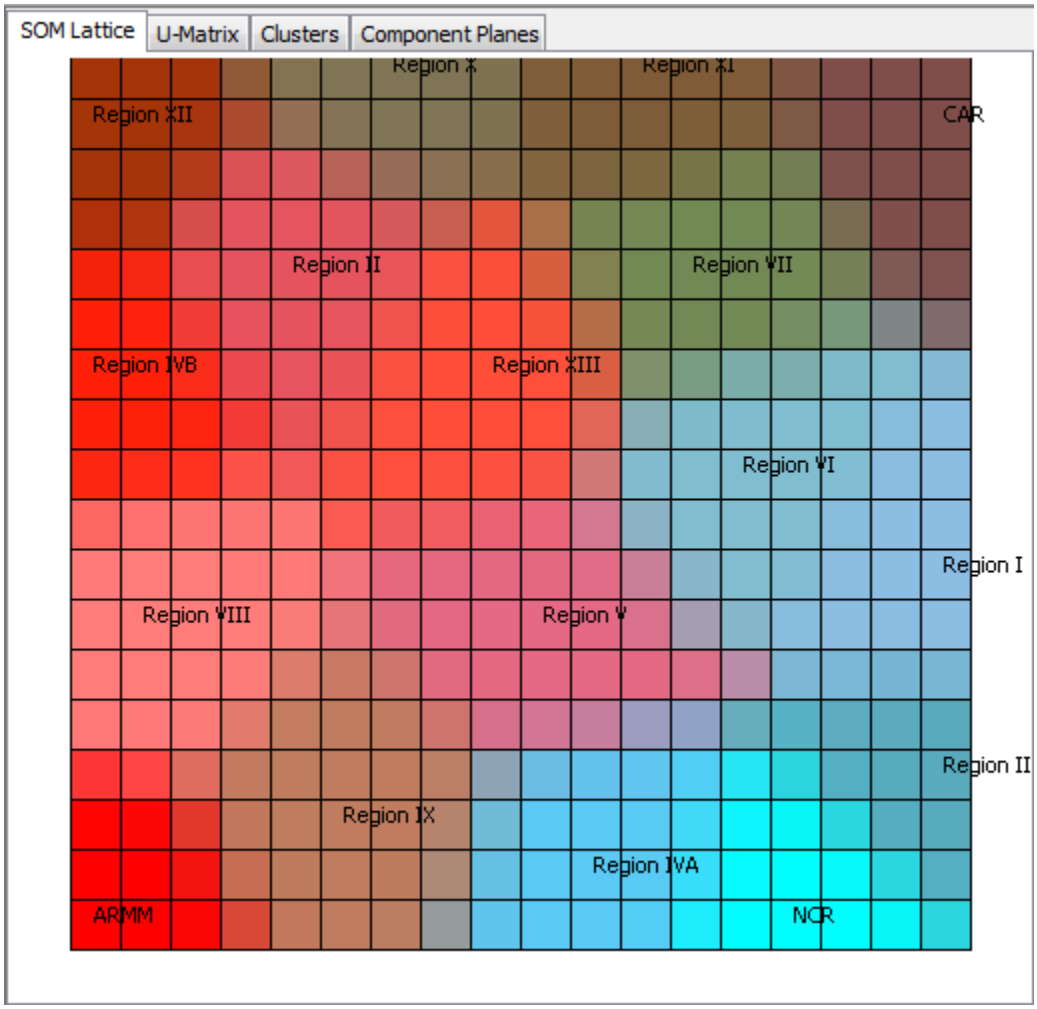

Figure 27: The initial SOM Lattice, SOM Visualize

The next tab, U-matrix, shows a landscape of the distances of the nodes from each other. To do so, the user must input a neighborhood size. The neighborhood size defines how many neighbors around should the node compare its values with. This lattice shows possible similarities and differences on the dataset. Which in turn, also shows possible clusters of data. Figure 28 shows the U-matrix visualization of the Philippine Regional Health Data. Regions I, III, IV-A, VI, and NCR are light colored which indicates possible similarities between these data entries. ARMM has a prominent black border that segregates it from the rest of the data which indicates that its variable values are very far from the others. This U-matrix used a neighborhood size of three.

The next tab is called the Cluster Tab. This tab shows the clusters formed upon

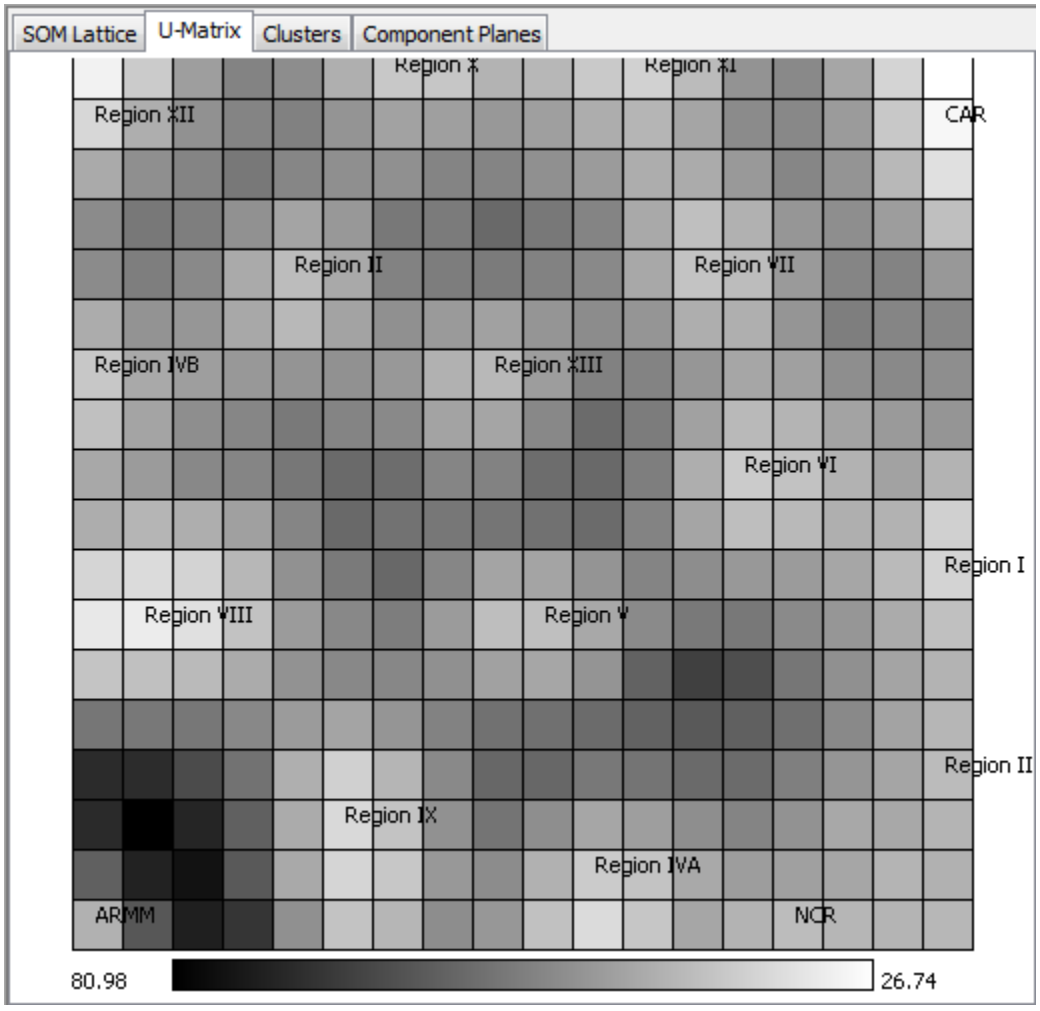

Figure 28: The U-matrix, SOM Visualize

running the k-means algorithm using the nodes. In using the cluster visualization, the user is required to input the desired number of clusters. The user may opt to believe the U-matrix when choosing the possible number of clusters. Figure 29 shows one cluster visualization of the Philippine Regional Health Data using three clusters. Almost similar to the U-matrix, the k-means algorithm clustered NCR, Regions I, III, IVA, VI but this time, including CAR. ARMM has formed its own cluster similar to what the U-matrix has given. This cluster was chosen by trying to run the algorithm thirty times and choosing the mode.

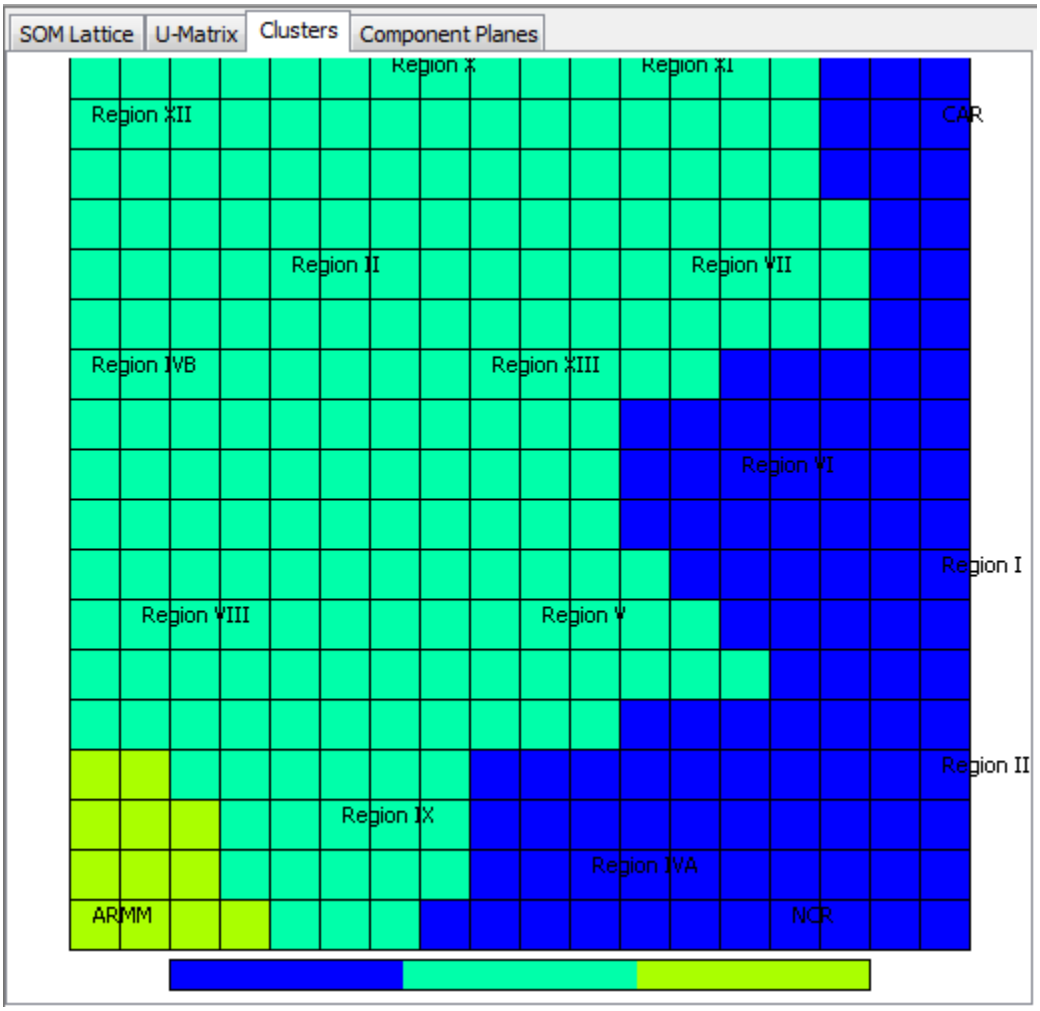

Figure 29: The Clusters, SOM Visualize

The last tab is the Component Plane Tab. The component plane tab is different from the three tabs in a sense that it is a collection of lattices that showcases a visualization for each variable indicated in the dataset. This part of the program should be able to give inferences and possible hypotheses that may be further tested statistically. Figure 30 shows a part of the component plane visualization of the Philippine Regional Health Data. By observation, it has been explained why ARMM has formed its own cluster. In most of the variable lattices its values were either extremely high or low in a negative sense. This shows that the health status of ARMM is relatively bad compared to the other regions. Also, the cluster that consists NCR, Regions I, III, IV-A, VI and CAR have also showed colors that are near each other in the spectrum indicating values that are near each other. Although,

in some of the lattices, particularly that of the fertility category. NCR has its extreme difference compared to other regions in a positive sense.

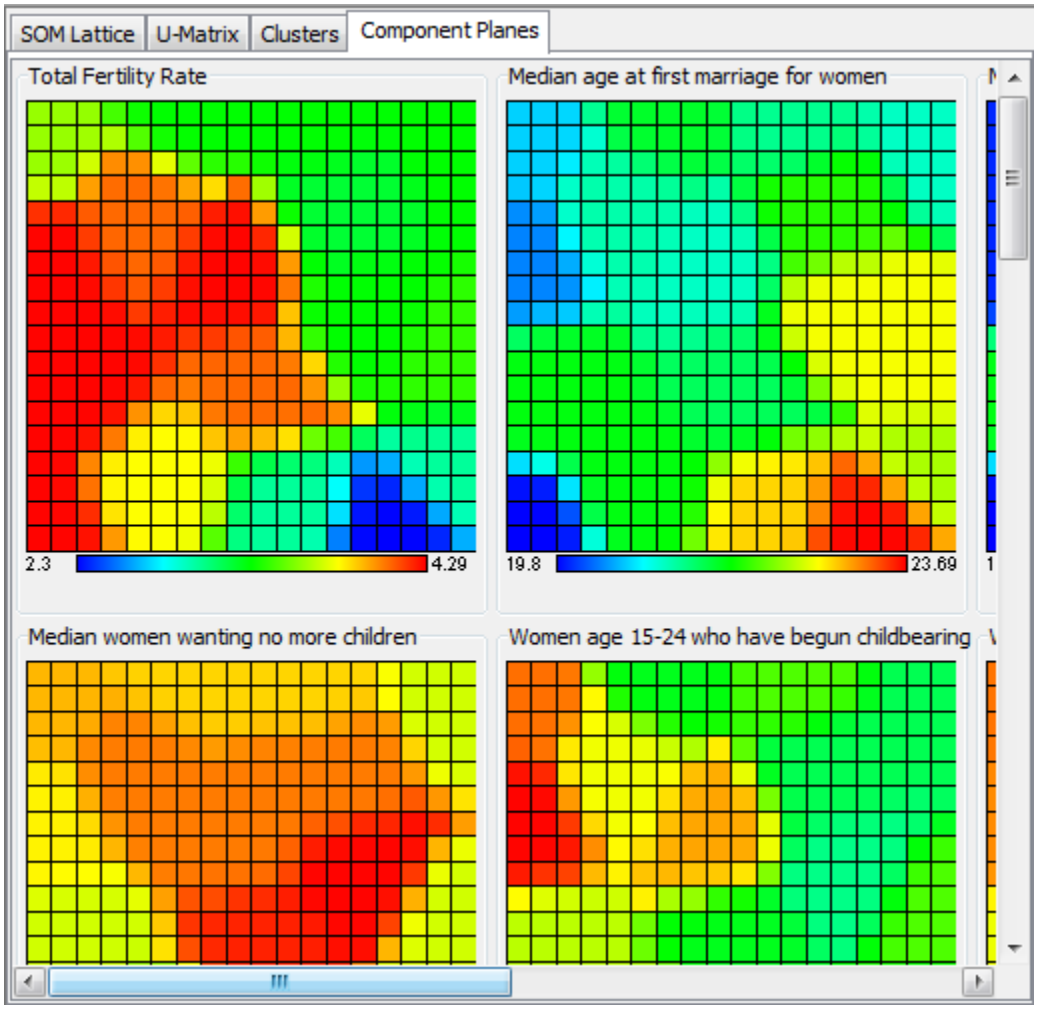

Figure 30: The Component Planes, SOM Visualize

Other features of the tool include the console panel, the file I/O, and the tutorial module.

The console panel is responsible of informing the user errors in data inputs, parameters, and the events that are currently happening with the tool. It also shows how much training has been done in fractions of ten whenever the tool is undergoing training. The console has three different color schemes. The red color scheme indicates that there is an error, the black color scheme is the scheme for updates, and information, and lastly the green color scheme indicates a successful task. Figure 31 shows the tool's console.

Figure 31: The Console Panel, SOM Visualize

The file I/O feature of the tool enables the user to save and load previous visualization files. With this feature, a user may also opt to pause in the middle of a training and save the file. This is suitable for users who do not want to leave their computers running when they are not around. Figure 32 shows the user locating for a SOM Visualize file (.vis).

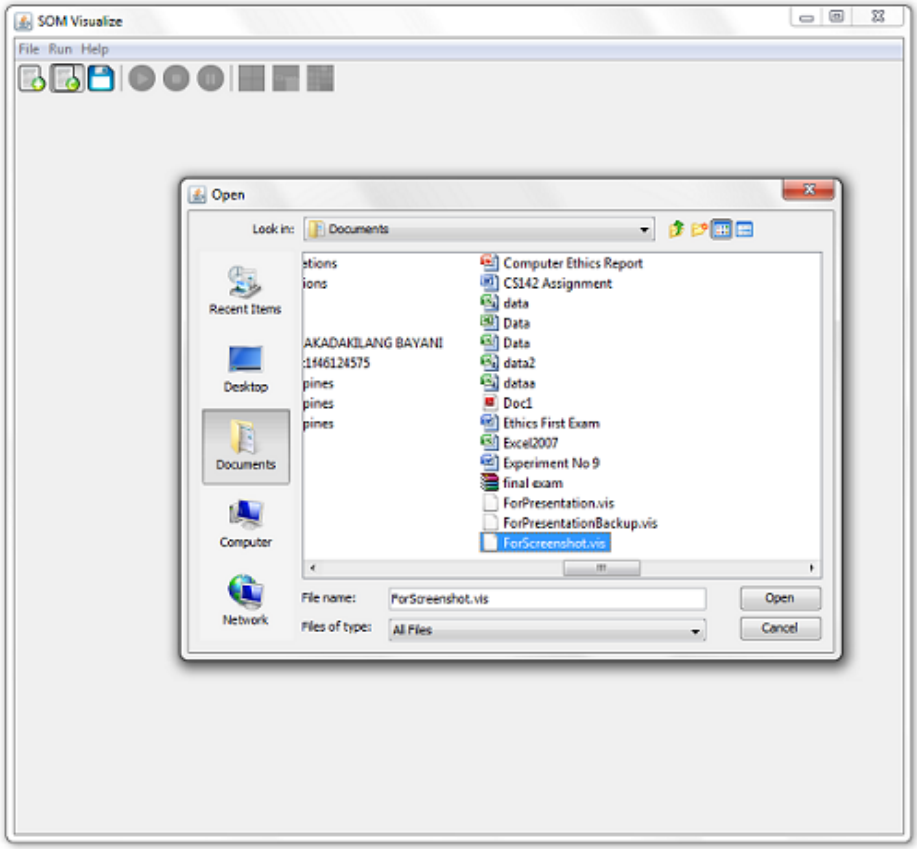

Figure 32: Choosing a SOM Visualize File, SOM Visualize

Figure 33 shows the tool that successfully loaded the file.

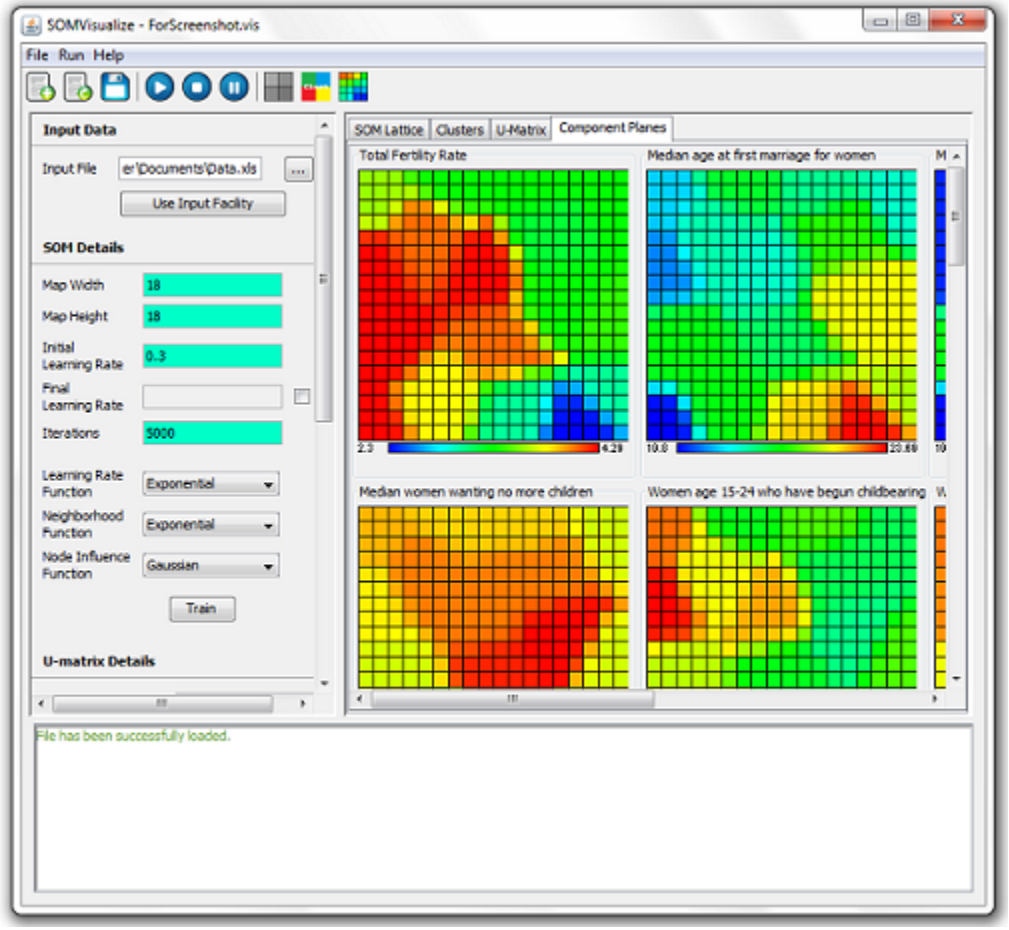

Figure 33: Successfully loading a Visualization File, SOM Visualize

The tool also has two browser-based modules, the GIS Visualizer and the Tutorial Module. The GIS Visualizer consists of two panels - the input panel and the result panel. In using the GIS Visualizer, the user must input an XML file generated by the tool. Thereafter, the GIS Visualizer shall parse the XML file and the user may be able to choose what variable to visualize on the map. Since the GIS Visualizer uses the Google Maps API, the user must be connected to the internet to visualize the data. Otherwise, the module will not work. Also, the visualizer stores the names and bounds of locations in a database. If the data label of a certain entry is not recognized by the map, no visualization shall be seen. Figure 34 shows the GIS Visualizer module visualizing "temp.xml" and the variable "Women currently using modern family planning method".

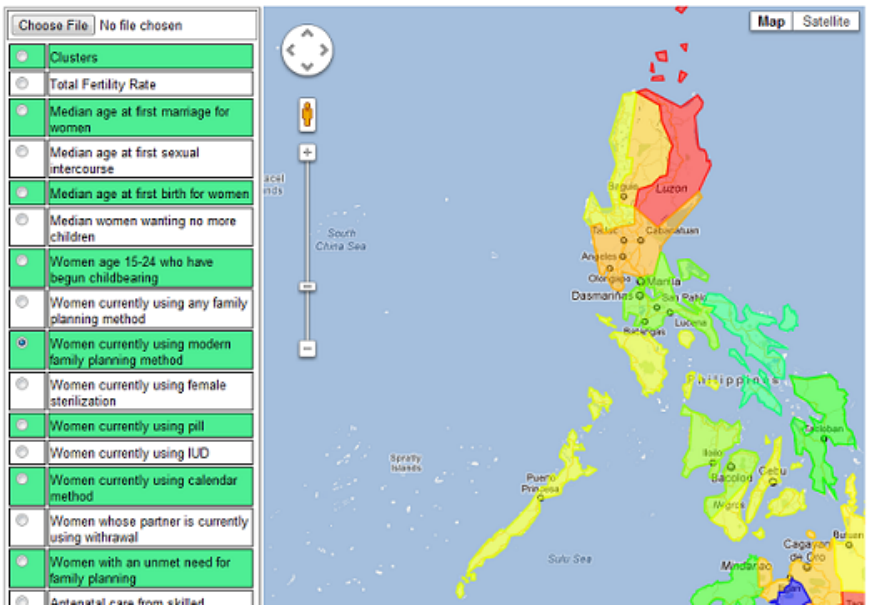

Figure 34: The GIS Visualizer, SOM Visualize

Finally, the tool also provides a tutorial on how to use it. The tutorial is a simple explanation on how to start using the tool, and the functionalities of the different panels. It also provides images to effectively give the users a grasp of what they should do. Figure 35 shows a part of the tutorial module.

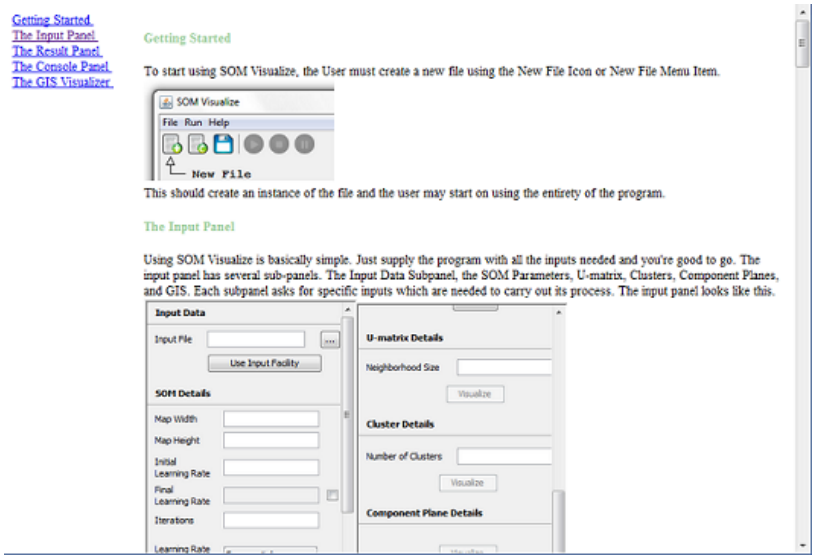

Figure 35: The tutorial module, SOM Visualize

The main window also consists of a toolbar of the commands such as train, pause, stop, and the visualization buttons. This provides quick access to certain commands.

VI. Discussion

SOM Visualize is a tool that allows for different visualizations of multidimensional data. The tool is interactive and straight-forward in nature which makes it user-friendly. All the user needs to do is supply an input data, and the map parameters. Through SOM Visualize we are able to see a fuzzy clustering of the data wherein data entries that have similarities are positioned near each other in the map. Because of this, we are able to analyze the data that are large in dimension.

SOM Visualize provides five visualization modules in total. This includes the initial SOM lattice, the Unified Distance Matrix, the Clusters, the Component Planes, and the GIS Visualization.

The first visualization module, the initial SOM lattice, gives the user an image of how the data entries are positioned in the map. With this, we are able to give the user an initial impression on whether the training was sufficient. Whenever the user feels contented with it, then he/she may choose to proceed with the other visualizations.

The second visualization module, the U-matrix visualization, provides a hint of the clustering of the data. It computes the node's distances from each other and give them colors on the grayscale color scheme. Through the U-matrix, we are able to detect which data entries are similar through each other and which are very different. The rule-of-thumb is data entries positioned near each other on the map that are light colored possibly forms a cluster while those with dark borders around them represent differences.

The third visualization module, the Cluster visualization, gives the user a visualization of the definite clusters that exist in the data through the k-means algorithm. The user may use the U-matrix to predict the number of clusters that exist in the lattice and use it as an input for the k-means algorithm. Based on the results gathered, the U-matrix proved to be an effective way to predict the number of clusters as they both suggest similar clusters.

The fourth visualization module, the Component Plane visualization, allows for comparisons between variables in the data. Using this visualization, the user may be given an idea of correlations that exist in the data space as well as assess the similarities that exist in each cluster of data given. It is also effective to use this type of visualization to look for

trends that exist on the data space.

Lastly, the GIS visualization module is an opportunity to be able to look at the visualization on a geographic map. Based on experience, trends and patterns that exist between geographic locations are easier to look at when placed on the geographic map than on the lattice. Thus, for geographic data, the GIS visualization is a big help.

SOM Visualize is an application mostly written in Java. Because of this, the tool is cross-platform and may be used so long as a Java Runtime Environment is installed on the computer. With this tool, we are able to see different visualizations and are able to give an initial analysis on data that are large in dimension. The tool gives a visualization on the clusters, correlations, trends, and patterns that exist in the data. Also through visualization, we are able to easily explain these trends. With these, the tool should be able to help researchers in fields that need multidimensional analysis.

However, tools like these always come together with advantages. It is known that the Self-Organizing Map is a computationally expensive algortihm. Thus, in order to use the tool, it is needed to run on computers with high specifications. Moreover, the tool has only implemented the fundamental visualizations done on the Self-Organizing Map. There are other visualizations that can be done for SOM and hence, we may consider the tool incomplete though relatively enough to perform data analysis. Also, there are other factors to consider such as proposed techniques to improve on the algorithm's accuracy and running time.

Visualization of data is an essential part in analyzing data especially when they are large in dimension. In the recent years, computational statistics has emerged to be an effective way to surpass obstacles such as these. Therefore, it is undeniable that the tool has opened opportunities for analysis and further improvements on the tool shall make it a powerful tool for visualization.

VII. Conclusion

SOM Visualize is a tool that uses the Self-Organizing Map algorithm for analysis of multi-dimensional data. Furthermore, it gives several visualization schemes such as the U-matrix, Clusters, and Component Planes to enable the discovery of subtle patterns in the dataset such as correlations, trends, and clusters. The tool is powerful considering its capabilities to handle noisy, skewed, and large datasets. It also enables viewing geographic data in a geographic map for spatial analysis and thus, we are able to gather new hypotheses that can be further tested statistically to discover new knowledge. SOM Visualize is interactive and user-friendly, making it a viable tool even for researches who are not very knowledgable about neural networks.

Through the Philippine Regional Health Data that has been processed using SOM Visualize. The tool's capabilities in uncovering patterns in multidimensional data has been proven. Therefore, we may consider the tool to be a success and may further be improved to allow for a better experience in Data Visualization.

VIII. Recommendations

Though SOM Visualize has successfully achieved its objectives, the tool has several functionalities that may further be improved. First, the tool so far has been using the rectangular topology of the lattice. However, some scholars say that the hexagonal topology would work better for some datasets since the distance from the node is equal throughout all of its neighbors. Second, the tool uses a static database of locations for the GIS Visualization. To allow for an interactive use, it is suggested that the tool should allow a user to add a specific location on the database together with its bounds and other information. Third, the tool has implemented only the fundamental visualizations for the lattice. There are several other visualizations that are capable of discovering other knowledge about the data. This should make the tool more versatile when it comes to analysis. Fourth, the visualization of the lattices are in a two-dimensional map, some scholars have suggested that the use of a three-dimensional visualization would be better to allow for an easy visualization of the clusters through divisions by peaks. Lastly, the use of k-means algorithm may not always be optimum for all datasets. Though it is not our main objective, it is suggested that the tool should allow the user on what algorithm will be used for choosing the clusters.

IX. Bibliography

[1] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth, "From data mining to knowledge discovery in databases," *American Association for Artificial Intelligence - AI Magazine*, pp. 37–54, 1997.

[2] M. Friendly, "Milestones in the history of thematic cartography, statistical graphics, and data visualization.," August 2009.

[3] T. Kohonen, "The self-organizing map," in *Proceedings of the IEEE*, vol. 78, September 1990.

[4] A. Ultsch and H. P. Siemon, "Kohonen's self organizing feature maps for exploratory data analysis," in *Intern. Neural Networks*, 1990.

[5] J. Vesanto and E. Alhoniemi, "Clustering of the self-organizing map," *IEEE Transactions on Neural Networks*, vol. 11, pp. 586–600, May 2000.

[6] T. Fincke, V. Lobo, and F. Bao, "Visualizing self-organizing maps with gis," in *GI Days*, 2008.

[7] K. E. Foote and M. Lynch, "Geographic information systems as an integrating technology: Context, concepts, and definitions."

[8] "Constitution of the world health organization - basic documents, forty-fifth edition," October 2006.

[9] T. Kohonen, "Self-organizing maps," *Springer-Verlag Berlin Heidelberg*, vol. 2, 1997.

[10] H. Basara and M. Yuan, "Community health assessment using self-organizing maps and geographic information systems," *Intenational Journal of Health Geographics*, vol. 7, pp. 67–75, 2008.

[11] E. L. Koua and M.-J. Kraak, "Geovisualization to support th exploration of large health and demographic survey data," *International Journal of Health Geographics*, vol. 3, pp. 12–24, 2004.

[12] E. L. Koua, "Using self-organizing maps for information visualization and knowledge discovery in complex geospatial datasets," in *Proceedings of the 21st International Cartographic Conference (ICC)*, pp. 1694–1702, 2003.

[13] A. Blazejewski and R. Coggins, "Application of self-organizing maps to clustering of high-frequency financial data," in *Conferences in Research and Practice in Information Technology* (J. Hogan, P. Montague, M. Purvis, and C. Steketee, eds.), vol. 32, 2004.

[14] P. Niemel and T. Honkela, "Analysis of parliamentary election results and socio-economic situation using self-organizing maps," in *Proceedings of the 7th International Workshop on Advances in Self-Organizing Maps*, Springer-Verlag, June 2009.

[15] M. Cottrell, P. Gaubert, C. Eloy, D. Franois, G. Hallaux, J. Lacaille, and M. Verleysen, "Fault prediction in aircraft engines using self-organizing maps," in *WSOM '09: Proceedings of the 7th International Workshop on Advances in Self-Organizing Maps*, June 2009.

[16] B. S. Penn, "Using self-organizing maps to visualize high-dimensional data," *Computers & Geosciences*, vol. 31, pp. 531–544, 2005.

[17] "Educause learning initiative - 7 things you should know about data visualization," October 2007.

[18] T. Germano, "Course on self organizing maps," March 1999.

[19] S. Marsland, *Machine Learning: An Algorithmic Perspective*. CRC Press, 2009.

[20] A. Ultsch, "U*-matrix: a tool to visualize clusters in high-dimensional data," 2003.

[21] K. Teknomo, "K-means clustering tutorials."

[22] "University of wyoming - basic gis concepts."

X. Appendix

SOMVisualize/Main.java

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package my.SOMVisualize;

import javax.swing.JOptionPane;

/**
 *
 * @author Mark Lester Ghany
 */
public class Main {
    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        /* Set the Windows look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (
            optional) ">
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.
                getInstalledLookAndFeels()) {
                if ("Windows".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {
            java.util.logging.Logger.getLogger(SOMVisualizeUI.class.getName()).log(
                java.util.logging.Level.SEVERE, null, ex);
        } catch (InstantiationException ex) {
            java.util.logging.Logger.getLogger(SOMVisualizeUI.class.getName()).log(
                java.util.logging.Level.SEVERE, null, ex);
        } catch (IllegalAccessException ex) {
            java.util.logging.Logger.getLogger(SOMVisualizeUI.class.getName()).log(
                java.util.logging.Level.SEVERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {
            java.util.logging.Logger.getLogger(SOMVisualizeUI.class.getName()).log(
                java.util.logging.Level.SEVERE, null, ex);
        }
        //</editor-fold>

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {

            @Override
            public void run() {
                new SOMVisualizeUI().setVisible(true);

            }
        });
    }
}
```

SOMVisualize/SOMVisualizeUI.java

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
```

```
/*
 * SOMVisualizeUI.java
 *
 * Created on Dec 2, 2011, 11:30:55 PM
 */
package my.SOMVisualize;

import java.awt.Color;
import java.io.*;
import java.util.*;
import java.util.logging.*;
import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import javax.swing.text.DefaultStyledDocument;
import javax.swing.text.StyledDocument;
import jxl.*;
import jxl.read.biff.BiffException;
/**
 *
 * @author Mark Lester
 */
public class SOMVisualizeUI extends javax.swing.JFrame {

    /** Creates new form SOMVisualizeUI */
    public SOMVisualizeUI() {
        initComponents();

        //added code
        inputPanel.setVisible(false);
        resultPanel.setVisible(false);
        consolePanel.setVisible(false);

        showStartUpScreen();

    }

    /** This method is called from within the constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is
     * always regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-BEGIN:
        initComponents
    private void initComponents() {

        jFileChooser1 = new javax.swing.JFileChooser();
        toolBar = new javax.swing.JToolBar();
        newFileButton = new javax.swing.JButton();
        openFileButton = new javax.swing.JButton();
        saveFileButton = new javax.swing.JButton();
        toolBarSeparator1 = new javax.swing.JToolBar.Separator();
        runButton = new javax.swing.JButton();
        stopButton = new javax.swing.JButton();
        pauseButton = new javax.swing.JButton();
        jSeparator1 = new javax.swing.JToolBar.Separator();
        umatButton = new javax.swing.JButton();
        clustButton = new javax.swing.JButton();
        compButton = new javax.swing.JButton();
        consolePanel = new javax.swing.JScrollPane();
        log = new javax.swing.JTextPane();
        inputPanel = new javax.swing.JScrollPane();
        inputPanelHolder = new javax.swing.JPanel();
        inputDataGroup = new javax.swing.JLabel();
        inputPanelSeparator1 = new javax.swing.JSeparator();
        inputFileLabel = new javax.swing.JLabel();
        inputFileTextField = new javax.swing.JTextField();
        browseToggleButton = new javax.swing.JButton();
```

```
inputFacilityToggleButton = new javax.swing.JButton();
somDetailsGroup = new javax.swing.JLabel();
inputPanelSeparator2 = new javax.swing.JSeparator();
mapWidthLabel = new javax.swing.JLabel();
mapWidthTextField = new javax.swing.JTextField();
mapHeightLabel = new javax.swing.JLabel();
mapHeightTextField = new javax.swing.JTextField();
initialLearningRateLabel = new javax.swing.JLabel();
initialLearningRateTextField = new javax.swing.JTextField();
finalLearningRateLabel = new javax.swing.JLabel();
finalLearningRateTextField = new javax.swing.JTextField();
finalLearningRateEnable = new javax.swing.JCheckBox();
iterationsLabel = new javax.swing.JLabel();
iterationsTextField = new javax.swing.JTextField();
learningRateFunctionLabel = new javax.swing.JLabel();
learningRateFunctionCombo = new javax.swing.JComboBox();
neighborhoodFunctionLabel = new javax.swing.JLabel();
neighborhoodFunctionCombo = new javax.swing.JComboBox();
nodeInfluenceFunctionLabel = new javax.swing.JLabel();
nodeInfluenceFunctionCombo = new javax.swing.JComboBox();
trainButton = new javax.swing.JButton();
uMatrixDetailsGroup = new javax.swing.JLabel();
inputPanelSeparator3 = new javax.swing.JSeparator();
neighborhoodSizeLabel = new javax.swing.JLabel();
neighborhoodSizeTextField = new javax.swing.JTextField();
uMatrixVisualizeButton = new javax.swing.JButton();
clusterDetailsGroup = new javax.swing.JLabel();
inputPanelSeparator4 = new javax.swing.JSeparator();
numberOfClustersLabel = new javax.swing.JLabel();
clusterVisualizeButton = new javax.swing.JButton();
componentPlaneDetailsGroup = new javax.swing.JLabel();
inputPanelSeparator5 = new javax.swing.JSeparator();
componentPlaneVisualizeButton = new javax.swing.JButton();
numberOfClustersTextField = new javax.swing.JTextField();
gisDetailsGroup = new javax.swing.JLabel();
inputPanelSeparator6 = new javax.swing.JSeparator();
gisVisualizeButton = new javax.swing.JButton();
iterationsClustersLabel = new javax.swing.JLabel();
iterationsClustersTextField = new javax.swing.JTextField();
clustersIdLabel = new javax.swing.JLabel();
clustersIdTextField = new javax.swing.JTextField();
resultPanel = new javax.swing.JTabbedPane();
jMenuBar1 = new javax.swing.JMenuBar();
fileMenu = new javax.swing.JMenu();
newFileMenu = new javax.swing.JMenuItem();
openFileMenu = new javax.swing.JMenuItem();
saveFileMenu = new javax.swing.JMenuItem();
saveAsFileMenu = new javax.swing.JMenuItem();
jMenu1 = new javax.swing.JMenu();
editMenu = new javax.swing.JMenu();
jMenuItem1 = new javax.swing.JMenuItem();
jFileChooser1.getAccessibleContext().setAccessibleDescription("");

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
setTitle("SOM Visualize");

toolBar.setFloatable(false);
toolBar.setRollover(true);

newFileButton.setIcon(new javax.swing.ImageIcon(getClass().getResource("/my/
    SOMVisualize/Icons/NewFileIcon.png"))); // NOI18N
newFileButton.setToolTipText("New File");
newFileButton.setFocusable(false);
newFileButton.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
newFileButton.setVerticalTextPosition(javax.swing.SwingConstants.BOTTOM);
newFileButton.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        newFileButtonActionPerformed(evt);
```

```
        }
    });
    toolBar.add(newFileButton);

    openFileButton.setIcon(new javax.swing.ImageIcon(getClass().getResource("/my/
        SOMVisualize/Icons/OpenFileIcon.png"))); // NOI18N
    openFileButton.setToolTipText("Open File");
    openFileButton.setFocusable(false);
    openFileButton.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
    openFileButton.setVerticalTextPosition(javax.swing.SwingConstants.BOTTOM);
    openFileButton.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            openFileButtonActionPerformed(evt);
        }
    });
    toolBar.add(openFileButton);

    saveFileButton.setIcon(new javax.swing.ImageIcon(getClass().getResource("/my/
        SOMVisualize/Icons/SaveFileIcon.png"))); // NOI18N
    saveFileButton.setToolTipText("Save File");
    saveFileButton.setFocusable(false);
    saveFileButton.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
    saveFileButton.setVerticalTextPosition(javax.swing.SwingConstants.BOTTOM);
    saveFileButton.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            saveFileButtonActionPerformed(evt);
        }
    });
    toolBar.add(saveFileButton);
    toolBar.add(toolBarSeparator1);

    runButton.setIcon(new javax.swing.ImageIcon(getClass().getResource("/my/
        SOMVisualize/Icons/RunIcon.png"))); // NOI18N
    runButton.setToolTipText("Run SOM Training");
    runButton.setEnabled(false);
    runButton.setFocusable(false);
    runButton.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
    runButton.setVerticalTextPosition(javax.swing.SwingConstants.BOTTOM);
    runButton.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            runButtonActionPerformed(evt);
        }
    });
    toolBar.add(runButton);

    stopButton.setIcon(new javax.swing.ImageIcon(getClass().getResource("/my/
        SOMVisualize/Icons/StopIcon.png"))); // NOI18N
    stopButton.setToolTipText("Stop SOM Training");
    stopButton.setEnabled(false);
    stopButton.setFocusable(false);
    stopButton.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
    stopButton.setVerticalTextPosition(javax.swing.SwingConstants.BOTTOM);
    stopButton.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            stopButtonActionPerformed(evt);
        }
    });
    toolBar.add(stopButton);

    pauseButton.setIcon(new javax.swing.ImageIcon(getClass().getResource("/my/
        SOMVisualize/Icons/PauseIcon.png"))); // NOI18N
    pauseButton.setToolTipText("Pause SOM Training");
    pauseButton.setEnabled(false);
    pauseButton.setFocusable(false);
    pauseButton.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
    pauseButton.setVerticalTextPosition(javax.swing.SwingConstants.BOTTOM);
    pauseButton.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
```

```
            pauseButtonActionPerformed(evt);
        }
    });
    toolBar.add(pauseButton);
    toolBar.add(jSeparator1);

    umatButton.setIcon(new javax.swing.ImageIcon(getClass().getResource("/my/
        SOMVisualize/Icons/UmatVis.png"))); // NOI18N
    umatButton.setToolTipText("Visualize U-matrix");
    umatButton.setEnabled(false);
    umatButton.setFocusable(false);
    umatButton.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
    umatButton.setVerticalTextPosition(javax.swing.SwingConstants.BOTTOM);
    umatButton.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            umatButtonActionPerformed(evt);
        }
    });
    toolBar.add(umatButton);

    clustButton.setIcon(new javax.swing.ImageIcon(getClass().getResource("/my/
        SOMVisualize/Icons/ClustVis.png"))); // NOI18N
    clustButton.setToolTipText("Visualize Clusters");
    clustButton.setEnabled(false);
    clustButton.setFocusable(false);
    clustButton.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
    clustButton.setVerticalTextPosition(javax.swing.SwingConstants.BOTTOM);
    clustButton.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            clustButtonActionPerformed(evt);
        }
    });
    toolBar.add(clustButton);

    compButton.setIcon(new javax.swing.ImageIcon(getClass().getResource("/my/
        SOMVisualize/Icons/CompVis.png"))); // NOI18N
    compButton.setToolTipText("Visualize Component Planes");
    compButton.setEnabled(false);
    compButton.setFocusable(false);
    compButton.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
    compButton.setVerticalTextPosition(javax.swing.SwingConstants.BOTTOM);
    compButton.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            compButtonActionPerformed(evt);
        }
    });
    toolBar.add(compButton);

    log.setBorder(javax.swing.BorderFactory.createBevelBorder(javax.swing.border.
        BevelBorder.LOWERED));
    consolePanel.setViewportView(log);

    inputPanel.setBorder(javax.swing.BorderFactory.createBevelBorder(javax.swing.
        border.BevelBorder.LOWERED));
    inputPanel.setToolTipText("");
    inputPanel.setInheritsPopupMenu(true);

    inputPanelHolder.setEnabled(false);

    inputDataGroup.setFont(new java.awt.Font("Tahoma", 1, 11));
    inputDataGroup.setText("Input Data");

    inputFileLabel.setText("Input File");

    inputFileTextField.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            inputFileTextFieldActionPerformed(evt);
        }
```

```
});

browseToggleButton.setText("...");
browseToggleButton.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        browseToggleButtonActionPerformed(evt);
    }
});

inputFacilityToggleButton.setText("Use Input Facility");
inputFacilityToggleButton.addActionListener(new java.awt.event.ActionListener
    () {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        inputFacilityToggleButtonActionPerformed(evt);
    }
});

somDetailsGroup.setFont(new java.awt.Font("Tahoma", 1, 11));
somDetailsGroup.setText("SOM Details");

mapWidthLabel.setText("Map Width");

mapWidthTextField.addCaretListener(new javax.swing.event.CaretListener() {
    public void caretUpdate(javax.swing.event.CaretEvent evt) {
        mapWidthTextFieldCaretUpdate(evt);
    }
});
mapWidthTextField.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        mapWidthTextFieldActionPerformed(evt);
    }
});

mapHeightLabel.setText("Map Height");

mapHeightTextField.addCaretListener(new javax.swing.event.CaretListener() {
    public void caretUpdate(javax.swing.event.CaretEvent evt) {
        mapHeightTextFieldCaretUpdate(evt);
    }
});

initialLearningRateLabel.setText("<html>\nInitial <br>\nLearning Rate\n</html
    >");

initialLearningRateTextField.addCaretListener(new javax.swing.event.
    CaretListener() {
    public void caretUpdate(javax.swing.event.CaretEvent evt) {
        initialLearningRateTextFieldCaretUpdate(evt);
    }
});

finalLearningRateLabel.setText("<html> Final <br> Learning Rate </html>");
finalLearningRateLabel.setEnabled(false);

finalLearningRateTextField.setEnabled(false);
finalLearningRateTextField.addCaretListener(new javax.swing.event.
    CaretListener() {
    public void caretUpdate(javax.swing.event.CaretEvent evt) {
        finalLearningRateTextFieldCaretUpdate(evt);
    }
});

finalLearningRateEnable.addActionListener(new java.awt.event.ActionListener()
     {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        finalLearningRateEnableActionPerformed(evt);
    }
});
```

```
iterationsLabel.setText("Iterations");

iterationsTextField.addCaretListener(new javax.swing.event.CaretListener() {
    public void caretUpdate(javax.swing.event.CaretEvent evt) {
        iterationsTextFieldCaretUpdate(evt);
    }
});

learningRateFunctionLabel.setText("<html> Learning Rate <br> Function</html>"
    );

learningRateFunctionCombo.setModel(new javax.swing.DefaultComboBoxModel(new
    String[] { "Exponential", "Power Series" }));
learningRateFunctionCombo.addActionListener(new java.awt.event.ActionListener
    () {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        learningRateFunctionComboActionPerformed(evt);
    }
});

neighborhoodFunctionLabel.setText("<html> Neighborhood <br> Function </html>"
    );

neighborhoodFunctionCombo.setModel(new javax.swing.DefaultComboBoxModel(new
    String[] { "Exponential" }));
neighborhoodFunctionCombo.addActionListener(new java.awt.event.ActionListener
    () {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        neighborhoodFunctionComboActionPerformed(evt);
    }
});

nodeInfluenceFunctionLabel.setText("<html> Node Influence <br> Function </
    html>");

nodeInfluenceFunctionCombo.setModel(new javax.swing.DefaultComboBoxModel(new
    String[] { "Gaussian" }));
nodeInfluenceFunctionCombo.addActionListener(new java.awt.event.
    ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        nodeInfluenceFunctionComboActionPerformed(evt);
    }
});

trainButton.setText("Train");
trainButton.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        trainButtonActionPerformed(evt);
    }
});

uMatrixDetailsGroup.setFont(new java.awt.Font("Tahoma", 1, 11));
uMatrixDetailsGroup.setText("U-matrix Details");

neighborhoodSizeLabel.setText("Neighborhood Size");

neighborhoodSizeTextField.addCaretListener(new javax.swing.event.
    CaretListener() {
    public void caretUpdate(javax.swing.event.CaretEvent evt) {
        neighborhoodSizeTextFieldCaretUpdate(evt);
    }
});

uMatrixVisualizeButton.setText("Visualize");
uMatrixVisualizeButton.setEnabled(false);
uMatrixVisualizeButton.addActionListener(new java.awt.event.ActionListener()
    {
```

```
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            uMatrixVisualizeButtonActionPerformed(evt);
        }
    });

    clusterDetailsGroup.setFont(new java.awt.Font("Tahoma", 1, 11));
    clusterDetailsGroup.setText("Cluster Details");

    numberOfClustersLabel.setText("Number of Clusters");

    clusterVisualizeButton.setText("Visualize");
    clusterVisualizeButton.setEnabled(false);
    clusterVisualizeButton.addActionListener(new java.awt.event.ActionListener()
        {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            clusterVisualizeButtonActionPerformed(evt);
        }
    });

    componentPlaneDetailsGroup.setFont(new java.awt.Font("Tahoma", 1, 11));
    componentPlaneDetailsGroup.setText("Component Plane Details");

    componentPlaneVisualizeButton.setText("Visualize");
    componentPlaneVisualizeButton.setEnabled(false);
    componentPlaneVisualizeButton.addActionListener(new java.awt.event.
        ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            componentPlaneVisualizeButtonActionPerformed(evt);
        }
    });

    numberOfClustersTextField.addCaretListener(new javax.swing.event.
        CaretListener() {
        public void caretUpdate(javax.swing.event.CaretEvent evt) {
            numberOfClustersTextFieldCaretUpdate(evt);
        }
    });

    gisDetailsGroup.setFont(new java.awt.Font("Tahoma", 1, 11));
    gisDetailsGroup.setText("GIS Details");

    gisVisualizeButton.setText("Visualize");
    gisVisualizeButton.setEnabled(false);
    gisVisualizeButton.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            gisVisualizeButtonActionPerformed(evt);
        }
    });

    iterationsClustersLabel.setText("Iterations");

    iterationsClustersTextField.addCaretListener(new javax.swing.event.
        CaretListener() {
        public void caretUpdate(javax.swing.event.CaretEvent evt) {
            iterationsClustersTextFieldCaretUpdate(evt);
        }
    });

    clustersIdLabel.setText("Cluster ID");

    clustersIdTextField.addCaretListener(new javax.swing.event.CaretListener() {
        public void caretUpdate(javax.swing.event.CaretEvent evt) {
            clustersIdTextFieldCaretUpdate(evt);
        }
    });
    clustersIdTextField.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            clustersIdTextFieldActionPerformed(evt);
```

```
    }
});

javax.swing.GroupLayout inputPanelHolderLayout = new javax.swing.GroupLayout(
    inputPanelHolder);
inputPanelHolder.setLayout(inputPanelHolderLayout);
inputPanelHolderLayout.setHorizontalGroup(
    inputPanelHolderLayout.createParallelGroup(javax.swing.GroupLayout.
        Alignment.LEADING)
    .addGroup(inputPanelHolderLayout.createSequentialGroup()
        .addGroup(inputPanelHolderLayout.createParallelGroup(javax.swing.
            GroupLayout.Alignment.LEADING)
            .addGroup(inputPanelHolderLayout.createSequentialGroup()
                .addGap(10, 10, 10)
                .addGroup(inputPanelHolderLayout.createParallelGroup(javax.
                    swing.GroupLayout.Alignment.LEADING)
                    .addComponent(neighborhoodSizeLabel)
                    .addComponent(numberOfClustersLabel)))
            .addGroup(inputPanelHolderLayout.createSequentialGroup()
                .addContainerGap()
                .addComponent(iterationsClustersLabel))
            .addGroup(inputPanelHolderLayout.createSequentialGroup()
                .addContainerGap()
                .addComponent(clustersIdLabel)))
        .addGap(18, 18, Short.MAX_VALUE)
        .addGroup(inputPanelHolderLayout.createParallelGroup(javax.swing.
            GroupLayout.Alignment.LEADING)
            .addComponent(clustersIdTextField, javax.swing.GroupLayout.
                PREFERRED_SIZE, 128, javax.swing.GroupLayout.PREFERRED_SIZE)
            .addComponent(iterationsClustersTextField, javax.swing.
                GroupLayout.DEFAULT_SIZE, 128, Short.MAX_VALUE))
        .addGap(19, 19, 19))
    .addGroup(inputPanelHolderLayout.createSequentialGroup()
        .addGap(120, 120, 120)
        .addComponent(neighborhoodSizeTextField, javax.swing.GroupLayout.
            DEFAULT_SIZE, 127, Short.MAX_VALUE)
        .addContainerGap(20, Short.MAX_VALUE))
    .addGroup(inputPanelHolderLayout.createSequentialGroup()
        .addGap(120, 120, 120)
        .addComponent(numberOfClustersTextField, javax.swing.GroupLayout.
            DEFAULT_SIZE, 128, Short.MAX_VALUE)
        .addGap(19, 19, 19))
    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
        inputPanelHolderLayout.createSequentialGroup()
        .addContainerGap(128, Short.MAX_VALUE)
        .addComponent(gisVisualizeButton)
        .addGap(66, 66, 66))
    .addGroup(inputPanelHolderLayout.createSequentialGroup()
        .addContainerGap()
        .addComponent(inputDataGroup)
        .addContainerGap(196, Short.MAX_VALUE))
    .addGroup(inputPanelHolderLayout.createSequentialGroup()
        .addContainerGap()
        .addComponent(somDetailsGroup)
        .addContainerGap(190, Short.MAX_VALUE))
    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
        inputPanelHolderLayout.createSequentialGroup()
        .addContainerGap()
        .addGroup(inputPanelHolderLayout.createParallelGroup(javax.swing.
            GroupLayout.Alignment.LEADING)
            .addComponent(nodeInfluenceFunctionLabel, javax.swing.GroupLayout
                .PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, javax.
                swing.GroupLayout.PREFERRED_SIZE)
            .addComponent(learningRateFunctionLabel, javax.swing.GroupLayout.
                PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, javax.
                swing.GroupLayout.PREFERRED_SIZE)
            .addComponent(neighborhoodFunctionLabel, javax.swing.GroupLayout.
                PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, javax.
```

```
                swing.GroupLayout.PREFERRED_SIZE)
            .addComponent(mapWidthLabel)
            .addComponent(mapHeightLabel)
            .addComponent(initialLearningRateLabel, javax.swing.GroupLayout.
                PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, javax.
                swing.GroupLayout.PREFERRED_SIZE)
            .addComponent(finalLearningRateLabel, javax.swing.GroupLayout.
                PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, javax.
                swing.GroupLayout.PREFERRED_SIZE)
            .addComponent(iterationsLabel))
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED
            )
        .addGroup(inputPanelHolderLayout.createParallelGroup(javax.swing.
            GroupLayout.Alignment.LEADING)
            .addComponent(mapWidthTextField, javax.swing.GroupLayout.
                Alignment.TRAILING, javax.swing.GroupLayout.DEFAULT_SIZE,
                128, Short.MAX_VALUE)
            .addComponent(initialLearningRateTextField, javax.swing.
                GroupLayout.Alignment.TRAILING, javax.swing.GroupLayout.
                DEFAULT_SIZE, 128, Short.MAX_VALUE)
            .addComponent(mapHeightTextField, javax.swing.GroupLayout.
                Alignment.TRAILING, javax.swing.GroupLayout.DEFAULT_SIZE,
                128, Short.MAX_VALUE)
            .addComponent(finalLearningRateTextField, javax.swing.GroupLayout
                .Alignment.TRAILING, javax.swing.GroupLayout.DEFAULT_SIZE,
                128, Short.MAX_VALUE)
            .addComponent(iterationsTextField, javax.swing.GroupLayout.
                Alignment.TRAILING, javax.swing.GroupLayout.DEFAULT_SIZE,
                128, Short.MAX_VALUE))
        .addGap(6, 6, 6)
        .addComponent(finalLearningRateEnable)
        .addGap(19, 19, 19))
    .addGroup(inputPanelHolderLayout.createSequentialGroup()
        .addGap(93, 93, 93)
        .addComponent(neighborhoodFunctionCombo, 0, 120, Short.MAX_VALUE)
        .addContainerGap(54, Short.MAX_VALUE))
    .addGroup(inputPanelHolderLayout.createSequentialGroup()
        .addGap(93, 93, 93)
        .addComponent(learningRateFunctionCombo, 0, 120, Short.MAX_VALUE)
        .addContainerGap(54, Short.MAX_VALUE))
    .addGroup(inputPanelHolderLayout.createSequentialGroup()
        .addContainerGap()
        .addComponent(uMatrixDetailsGroup)
        .addContainerGap(164, Short.MAX_VALUE))
    .addGroup(inputPanelHolderLayout.createSequentialGroup()
        .addGap(93, 93, 93)
        .addComponent(nodeInfluenceFunctionCombo, 0, 120, Short.MAX_VALUE)
        .addContainerGap(54, Short.MAX_VALUE))
    .addGroup(inputPanelHolderLayout.createSequentialGroup()
        .addContainerGap()
        .addComponent(clusterDetailsGroup)
        .addContainerGap(175, Short.MAX_VALUE))
    .addGroup(javax.swing.GroupLayout.Alignment.CENTER,
        inputPanelHolderLayout.createSequentialGroup()
        .addGap(73, 73, 73)
        .addComponent(inputFacilityToggleButton, javax.swing.GroupLayout.
            DEFAULT_SIZE, 152, Short.MAX_VALUE)
        .addGap(42, 42, 42))
    .addGroup(javax.swing.GroupLayout.Alignment.CENTER,
        inputPanelHolderLayout.createSequentialGroup()
        .addGap(119, 119, 119)
        .addComponent(trainButton)
        .addGap(89, 89, 89))
    .addGroup(javax.swing.GroupLayout.Alignment.CENTER,
        inputPanelHolderLayout.createSequentialGroup()
        .addGap(111, 111, 111)
        .addComponent(uMatrixVisualizeButton)
        .addGap(81, 81, 81))
```

```
        .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
            inputPanelHolderLayout.createSequentialGroup()
            .addGroup(inputPanelHolderLayout.createParallelGroup(javax.swing.
                GroupLayout.Alignment.TRAILING)
                .addComponent(inputPanelSeparator1, javax.swing.GroupLayout.
                    Alignment.LEADING, javax.swing.GroupLayout.DEFAULT_SIZE, 248,
                     Short.MAX_VALUE)
                .addComponent(inputPanelSeparator2, javax.swing.GroupLayout.
                    Alignment.LEADING, javax.swing.GroupLayout.DEFAULT_SIZE, 248,
                     Short.MAX_VALUE)
                .addGroup(inputPanelHolderLayout.createSequentialGroup()
                    .addGroup(inputPanelHolderLayout.createParallelGroup(javax.
                        swing.GroupLayout.Alignment.LEADING)
                        .addGroup(inputPanelHolderLayout.createSequentialGroup()
                            .addContainerGap()
                            .addComponent(inputFileLabel)
                            .addGap(22, 22, 22))
                        .addGroup(inputPanelHolderLayout.createSequentialGroup()
                            .addGap(73, 73, 73)
                            .addComponent(inputFileTextField, javax.swing.
                                GroupLayout.DEFAULT_SIZE, 131, Short.MAX_VALUE)))
                    .addGap(18, 18, 18)
                    .addComponent(browseToggleButton, javax.swing.GroupLayout.
                        PREFERRED_SIZE, 26, javax.swing.GroupLayout.
                        PREFERRED_SIZE)))
            .addGap(19, 19, 19))
        .addGroup(inputPanelHolderLayout.createSequentialGroup()
            .addComponent(inputPanelSeparator3, javax.swing.GroupLayout.
                DEFAULT_SIZE, 247, Short.MAX_VALUE)
            .addContainerGap(20, Short.MAX_VALUE))
        .addGroup(inputPanelHolderLayout.createSequentialGroup()
            .addGap(9, 9, 9)
            .addComponent(gisDetailsGroup)
            .addContainerGap(196, javax.swing.GroupLayout.PREFERRED_SIZE))
        .addGroup(inputPanelHolderLayout.createSequentialGroup()
            .addComponent(componentPlaneDetailsGroup)
            .addContainerGap(126, javax.swing.GroupLayout.PREFERRED_SIZE))
        .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
            inputPanelHolderLayout.createSequentialGroup()
            .addContainerGap(128, Short.MAX_VALUE)
            .addGroup(inputPanelHolderLayout.createParallelGroup(javax.swing.
                GroupLayout.Alignment.TRAILING)
                .addComponent(clusterVisualizeButton)
                .addComponent(componentPlaneVisualizeButton))
            .addGap(66, 66, 66))
        .addGroup(inputPanelHolderLayout.createSequentialGroup()
            .addGroup(inputPanelHolderLayout.createParallelGroup(javax.swing.
                GroupLayout.Alignment.LEADING)
                .addComponent(inputPanelSeparator4, javax.swing.GroupLayout.
                    Alignment.TRAILING, javax.swing.GroupLayout.DEFAULT_SIZE,
                    257, Short.MAX_VALUE)
                .addComponent(inputPanelSeparator5, javax.swing.GroupLayout.
                    Alignment.TRAILING, javax.swing.GroupLayout.DEFAULT_SIZE,
                    257, Short.MAX_VALUE)
                .addComponent(inputPanelSeparator6, javax.swing.GroupLayout.
                    Alignment.TRAILING, javax.swing.GroupLayout.DEFAULT_SIZE,
                    257, Short.MAX_VALUE))
            .addContainerGap())
);
inputPanelHolderLayout.setVerticalGroup(
    inputPanelHolderLayout.createParallelGroup(javax.swing.GroupLayout.
        Alignment.LEADING)
    .addGroup(inputPanelHolderLayout.createSequentialGroup()
        .addGap(6, 6, 6)
        .addComponent(inputDataGroup)
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addComponent(inputPanelSeparator1, javax.swing.GroupLayout.
            PREFERRED_SIZE, 2, javax.swing.GroupLayout.PREFERRED_SIZE)
```

```
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
.addGroup(inputPanelHolderLayout.createParallelGroup(javax.swing.
    GroupLayout.Alignment.BASELINE, false)
    .addComponent(inputFileTextField, javax.swing.GroupLayout.
        PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, javax.
        swing.GroupLayout.PREFERRED_SIZE)
    .addComponent(inputFileLabel)
    .addComponent(browseToggleButton))
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
.addComponent(inputFacilityToggleButton)
.addGap(18, 18, 18)
.addComponent(somDetailsGroup)
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
.addComponent(inputPanelSeparator2, javax.swing.GroupLayout.
    PREFERRED_SIZE, 2, javax.swing.GroupLayout.PREFERRED_SIZE)
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
.addGroup(inputPanelHolderLayout.createParallelGroup(javax.swing.
    GroupLayout.Alignment.CENTER)
    .addComponent(mapWidthLabel)
    .addComponent(mapWidthTextField, javax.swing.GroupLayout.
        PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, javax.
        swing.GroupLayout.PREFERRED_SIZE))
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
.addGroup(inputPanelHolderLayout.createParallelGroup(javax.swing.
    GroupLayout.Alignment.CENTER)
    .addComponent(mapHeightLabel)
    .addComponent(mapHeightTextField, javax.swing.GroupLayout.
        PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, javax.
        swing.GroupLayout.PREFERRED_SIZE))
.addGap(9, 9, 9)
.addGroup(inputPanelHolderLayout.createParallelGroup(javax.swing.
    GroupLayout.Alignment.CENTER)
    .addComponent(initialLearningRateLabel, javax.swing.GroupLayout.
        PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, javax.
        swing.GroupLayout.PREFERRED_SIZE)
    .addComponent(initialLearningRateTextField, javax.swing.
        GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.
        DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
.addGroup(inputPanelHolderLayout.createParallelGroup(javax.swing.
    GroupLayout.Alignment.CENTER)
    .addComponent(finalLearningRateLabel, javax.swing.GroupLayout.
        PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, javax.
        swing.GroupLayout.PREFERRED_SIZE)
    .addComponent(finalLearningRateTextField, javax.swing.GroupLayout
        .PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, javax.
        swing.GroupLayout.PREFERRED_SIZE)
    .addComponent(finalLearningRateEnable))
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
.addGroup(inputPanelHolderLayout.createParallelGroup(javax.swing.
    GroupLayout.Alignment.CENTER)
    .addComponent(iterationsLabel)
    .addComponent(iterationsTextField, javax.swing.GroupLayout.
        PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, javax.
        swing.GroupLayout.PREFERRED_SIZE))
.addGap(18, 18, 18)
.addGroup(inputPanelHolderLayout.createParallelGroup(javax.swing.
    GroupLayout.Alignment.CENTER)
    .addComponent(learningRateFunctionLabel, javax.swing.GroupLayout.
        PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, javax.
        swing.GroupLayout.PREFERRED_SIZE)
    .addComponent(learningRateFunctionCombo, javax.swing.GroupLayout.
        PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, javax.
        swing.GroupLayout.PREFERRED_SIZE))
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
.addGroup(inputPanelHolderLayout.createParallelGroup(javax.swing.
    GroupLayout.Alignment.CENTER)
```

```
            .addComponent(neighborhoodFunctionCombo, javax.swing.GroupLayout.
                PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, javax.
                swing.GroupLayout.PREFERRED_SIZE)
            .addComponent(neighborhoodFunctionLabel, javax.swing.GroupLayout.
                PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, javax.
                swing.GroupLayout.PREFERRED_SIZE))
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addGroup(inputPanelHolderLayout.createParallelGroup(javax.swing.
            GroupLayout.Alignment.CENTER)
            .addComponent(nodeInfluenceFunctionCombo, javax.swing.GroupLayout
                .PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, javax.
                swing.GroupLayout.PREFERRED_SIZE)
            .addComponent(nodeInfluenceFunctionLabel, javax.swing.GroupLayout
                .PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, javax.
                swing.GroupLayout.PREFERRED_SIZE))
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED
            )
        .addComponent(trainButton)
        .addGap(25, 25, 25)
        .addComponent(uMatrixDetailsGroup)
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addComponent(inputPanelSeparator3, javax.swing.GroupLayout.
            PREFERRED_SIZE, 2, javax.swing.GroupLayout.PREFERRED_SIZE)
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED
            )
        .addGroup(inputPanelHolderLayout.createParallelGroup(javax.swing.
            GroupLayout.Alignment.CENTER)
            .addComponent(neighborhoodSizeTextField, javax.swing.GroupLayout.
                PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, javax.
                swing.GroupLayout.PREFERRED_SIZE)
            .addComponent(neighborhoodSizeLabel))
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED
            )
        .addComponent(uMatrixVisualizeButton)
        .addGap(18, 18, 18)
        .addComponent(clusterDetailsGroup)
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addComponent(inputPanelSeparator4, javax.swing.GroupLayout.
            PREFERRED_SIZE, 10, javax.swing.GroupLayout.PREFERRED_SIZE)
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addGroup(inputPanelHolderLayout.createParallelGroup(javax.swing.
            GroupLayout.Alignment.BASELINE)
            .addComponent(numberOfClustersLabel)
            .addComponent(numberOfClustersTextField, javax.swing.GroupLayout.
                PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, javax.
                swing.GroupLayout.PREFERRED_SIZE))
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addGroup(inputPanelHolderLayout.createParallelGroup(javax.swing.
            GroupLayout.Alignment.BASELINE)
            .addComponent(iterationsClustersLabel)
            .addComponent(iterationsClustersTextField, javax.swing.
                GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.
                DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
        .addGap(18, 18, 18)
        .addComponent(clusterVisualizeButton)
        .addGap(18, 18, 18)
        .addComponent(componentPlaneDetailsGroup)
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addComponent(inputPanelSeparator5, javax.swing.GroupLayout.
            PREFERRED_SIZE, 7, javax.swing.GroupLayout.PREFERRED_SIZE)
        .addGap(18, 18, 18)
        .addComponent(componentPlaneVisualizeButton)
        .addGap(18, 18, 18)
        .addComponent(gisDetailsGroup)
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addComponent(inputPanelSeparator6, javax.swing.GroupLayout.
            PREFERRED_SIZE, 10, javax.swing.GroupLayout.PREFERRED_SIZE)
        .addGap(15, 15, 15)
```

```
            .addGroup(inputPanelHolderLayout.createParallelGroup(javax.swing.
                GroupLayout.Alignment.BASELINE)
                .addComponent(clustersIdLabel)
                .addComponent(clustersIdTextField, javax.swing.GroupLayout.
                    PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, javax.
                    swing.GroupLayout.PREFERRED_SIZE))
            .addGap(18, 18, 18)
            .addComponent(gisVisualizeButton)
            .addContainerGap())
);

mapWidthTextField.getAccessibleContext().setAccessibleName("");

inputPanel.setViewportView(inputPanelHolder);

resultPanel.setBorder(javax.swing.BorderFactory.createBevelBorder(javax.swing
    .border.BevelBorder.LOWERED));

fileMenu.setText("File");

newFileMenu.setAccelerator(javax.swing.KeyStroke.getKeyStroke(java.awt.event.
    KeyEvent.VK_N, java.awt.event.InputEvent.CTRL_MASK));
newFileMenu.setIcon(new javax.swing.ImageIcon(getClass().getResource("/my/
    SOMVisualize/Icons/NewFileIcon.png"))); // NOI18N
newFileMenu.setText("New File");
newFileMenu.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        newFileMenuActionPerformed(evt);
    }
});
fileMenu.add(newFileMenu);

openFileMenu.setAccelerator(javax.swing.KeyStroke.getKeyStroke(java.awt.event
    .KeyEvent.VK_O, java.awt.event.InputEvent.CTRL_MASK));
openFileMenu.setIcon(new javax.swing.ImageIcon(getClass().getResource("/my/
    SOMVisualize/Icons/OpenFileIcon.png"))); // NOI18N
openFileMenu.setText("Open");
openFileMenu.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        openFileMenuActionPerformed(evt);
    }
});
fileMenu.add(openFileMenu);

saveFileMenu.setAccelerator(javax.swing.KeyStroke.getKeyStroke(java.awt.event
    .KeyEvent.VK_S, java.awt.event.InputEvent.CTRL_MASK));
saveFileMenu.setIcon(new javax.swing.ImageIcon(getClass().getResource("/my/
    SOMVisualize/Icons/SaveFileIcon.png"))); // NOI18N
saveFileMenu.setText("Save");
fileMenu.add(saveFileMenu);

saveAsFileMenu.setAccelerator(javax.swing.KeyStroke.getKeyStroke(java.awt.
    event.KeyEvent.VK_F12, 0));
saveAsFileMenu.setText("Save as...");
saveAsFileMenu.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        saveAsFileMenuActionPerformed(evt);
    }
});
fileMenu.add(saveAsFileMenu);

jMenuBar1.add(fileMenu);

jMenu1.setText("Run");
jMenuBar1.add(jMenu1);

editMenu.setText("Help");
```

```
        jMenuItem1.setAccelerator(javax.swing.KeyStroke.getKeyStroke(java.awt.event.
            KeyEvent.VK_F1, 0));
        jMenuItem1.setText("View Tutorial");
        jMenuItem1.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jMenuItem1ActionPerformed(evt);
            }
        });
        editMenu.add(jMenuItem1);

        jMenuBar1.add(editMenu);

        setJMenuBar(jMenuBar1);

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane()
            );
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.
                    Alignment.LEADING)
                    .addGroup(layout.createSequentialGroup()
                        .addContainerGap()
                        .addComponent(consolePanel, javax.swing.GroupLayout.
                            DEFAULT_SIZE, 782, Short.MAX_VALUE))
                    .addGroup(layout.createSequentialGroup()
                        .addGap(6, 6, 6)
                        .addComponent(inputPanel, javax.swing.GroupLayout.
                            PREFERRED_SIZE, 256, javax.swing.GroupLayout.
                            PREFERRED_SIZE)
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.
                            RELATED)
                        .addComponent(resultPanel, javax.swing.GroupLayout.
                            DEFAULT_SIZE, 524, Short.MAX_VALUE))
                    .addComponent(toolBar, javax.swing.GroupLayout.PREFERRED_SIZE,
                        359, javax.swing.GroupLayout.PREFERRED_SIZE))
                .addContainerGap())
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addComponent(toolBar, javax.swing.GroupLayout.PREFERRED_SIZE, 30,
                    javax.swing.GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.
                    Alignment.LEADING)
                    .addComponent(inputPanel, javax.swing.GroupLayout.DEFAULT_SIZE,
                        503, Short.MAX_VALUE)
                    .addComponent(resultPanel, javax.swing.GroupLayout.DEFAULT_SIZE,
                        503, Short.MAX_VALUE))
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(consolePanel, javax.swing.GroupLayout.PREFERRED_SIZE,
                    140, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap())
        );

        pack();
    }// </editor-fold>//GEN-END:initComponents


private void newFileMenuActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:
    event_newFileMenuActionPerformed

    //always ensure that there is no current file running else, ask the user if he
        wants to terminate
    int choice = 0;
```

```
if(visFileExists){
    choice = 1;
    Object[] options = {"Yes", "No"};
    choice = JOptionPane.showOptionDialog(this,
        "There is a file currently running. Are you sure you want to create a new
            file?",
        "Warning",
        JOptionPane.YES_NO_OPTION,
        JOptionPane.WARNING_MESSAGE,
        null,
        options,
        options[1]);
}

if(choice == 0){
    //assume there is a file and show everything
    this.setTitle("SOM Visualize - Untitled");

    this.inputPanel.setVisible(true);
    this.resultPanel.setVisible(true);
    this.consolePanel.setVisible(true);

    this.resultPanel.setVisible(false);

    //reset all elements

    /*RESULT*/
    resultPanel.removeAll();

    /*CONSOLE*/
    log.setText("");
    doc = new DefaultStyledDocument();
    cl = new ConsoleLog(log, doc);
    consolePanel.setAutoscrolls(true);

    /*INPUT*/
    this.inputFileTextField.setText("");
    this.mapHeightTextField.setText("");
    this.mapWidthTextField.setText("");
    this.initialLearningRateTextField.setText("");
    this.finalLearningRateTextField.setText("");
    this.iterationsTextField.setText("");
    this.learningRateFunctionCombo.setSelectedIndex(0);
    this.neighborhoodFunctionCombo.setSelectedIndex(0);
    this.nodeInfluenceFunctionCombo.setSelectedIndex(0);

    this.neighborhoodSizeTextField.setText("");
    this.numberOfClustersTextField.setText("");

    //buttons
    this.uMatrixVisualizeButton.setEnabled(false);
    this.clusterVisualizeButton.setEnabled(false);
    this.componentPlaneVisualizeButton.setEnabled(false);
    this.gisVisualizeButton.setEnabled(false);

    //File input and input data
    excelFile = null;
    inputDataClass = null;

    //Trainer and visualizers
    trainer = null;
    umat = null;
    clust = null;
    comp = null;

    //Miscellaneous
    maxMin = null;
```

```
        //Save File
        saveFile = null;

        //Flags
        umatDone = false;
        compDone = false;

        //inputFacility
        inputFac = null;

        this.visFileExists = true;

        runButton.setEnabled(true);
        pauseButton.setEnabled(true);
        stopButton.setEnabled(true);
    }
}//GEN-LAST:event_newFileMenuActionPerformed

private void newFileButtonActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
    FIRST:event_newFileButtonActionPerformed
    this.newFileMenu.doClick();
}//GEN-LAST:event_newFileButtonActionPerformed

private void saveFileButtonActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
    FIRST:event_saveFileButtonActionPerformed
    if(visFileExists){
        if(saveFile == null){
            javax.swing.JFileChooser fc = new javax.swing.JFileChooser();
            int returnVal = fc.showSaveDialog(null);

            if(returnVal == JFileChooser.APPROVE_OPTION){
                saveFile = fc.getSelectedFile();
            }
        }

        //check if the file exists now, then save details to file
        try {
            String fileName;
            if(!(saveFile.getAbsolutePath().contains(".vis"))){
                fileName = saveFile.getAbsolutePath() + ".vis";
            }
            else{
                fileName = saveFile.getAbsolutePath();
            }
            ObjectOutputStream out = new ObjectOutputStream(new FileOutputStream(
                fileName));

            /*WRITE NECESSARY DATA FROM INPUT PANEL*/
            out.writeObject(inputFileTextField.getText());
            out.writeObject(mapWidthTextField.getText());
            out.writeObject(mapHeightTextField.getText());
            out.writeObject(initialLearningRateTextField.getText());
            out.writeObject(finalLearningRateTextField.getText());
            out.writeObject(iterationsTextField.getText());
            out.writeObject(learningRateFunctionCombo.getSelectedIndex());
            out.writeObject(neighborhoodFunctionCombo.getSelectedIndex());
            out.writeObject(nodeInfluenceFunctionCombo.getSelectedIndex());

            out.writeObject(neighborhoodSizeTextField.getText());
            out.writeObject(numberOfClustersTextField.getText());

            out.writeObject(trainer);
            out.writeObject(clust);

            out.writeObject(inputDataClass);
            out.writeObject(maxMin);

            out.writeObject(umatDone);
```

```
                out.writeObject(compDone);

                out.close();

                this.setTitle("SOMVisualize - " + saveFile.getName());

            } catch (IOException ex) {
                Logger.getLogger(SOMVisualizeUI.class.getName()).log(Level.SEVERE, null,
                    ex);
            }
        }


    }//GEN-LAST:event_saveFileButtonActionPerformed

    private void stopButtonActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:
        event_stopButtonActionPerformed
        if(mapTrainer.isAlive()){
            cl.throwErrorMessage("Training has been aborted!\n");
        }
        trainer.stop();
        while(mapTrainer.isAlive()){}
        trainer = null;
        mapTrainer = null;

    }//GEN-LAST:event_stopButtonActionPerformed

    private void mapWidthTextFieldActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
        FIRST:event_mapWidthTextFieldActionPerformed
    // TODO add your handling code here:
    }//GEN-LAST:event_mapWidthTextFieldActionPerformed

    private void inputFileTextFieldActionPerformed(java.awt.event.ActionEvent evt) {//GEN
        -FIRST:event_inputFileTextFieldActionPerformed

    }//GEN-LAST:event_inputFileTextFieldActionPerformed

    private void finalLearningRateEnableActionPerformed(java.awt.event.ActionEvent evt) {
        //GEN-FIRST:event_finalLearningRateEnableActionPerformed
        if (finalLearningRateEnable.isSelected()){
            finalLearningRateTextField.setEnabled(true);
            finalLearningRateLabel.setEnabled(true);
        }
        else{
            finalLearningRateTextField.setEnabled(false);
            finalLearningRateLabel.setEnabled(false);
        }
    }//GEN-LAST:event_finalLearningRateEnableActionPerformed

    private void learningRateFunctionComboActionPerformed(java.awt.event.ActionEvent evt)
         {//GEN-FIRST:event_learningRateFunctionComboActionPerformed

    }//GEN-LAST:event_learningRateFunctionComboActionPerformed

    private void neighborhoodFunctionComboActionPerformed(java.awt.event.ActionEvent evt)
         {//GEN-FIRST:event_neighborhoodFunctionComboActionPerformed
    // TODO add your handling code here:
    }//GEN-LAST:event_neighborhoodFunctionComboActionPerformed

    private void nodeInfluenceFunctionComboActionPerformed(java.awt.event.ActionEvent evt
        ) {//GEN-FIRST:event_nodeInfluenceFunctionComboActionPerformed
    // TODO add your handling code here:
    }//GEN-LAST:event_nodeInfluenceFunctionComboActionPerformed

    private void trainButtonActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:
        event_trainButtonActionPerformed

        /* Get input data details
```

```
 * For now, let it be hard coded
 * If training input is incomplete do not proceed
 */
int mapWidth;
int mapHeight;
double initialLearningRate;
double finalLearningRate;
int iterations;

int learningRateFunction;
int neighborhoodFunction;
int nodeInfluenceFunction;


try{
    mapWidth = Integer.parseInt(mapWidthTextField.getText());
    mapHeight = Integer.parseInt(mapHeightTextField.getText());
    initialLearningRate = Double.parseDouble(initialLearningRateTextField.getText
        ());
    if(finalLearningRateEnable.isSelected()){
        finalLearningRate = Double.parseDouble(finalLearningRateTextField.getText
            ());
    }
    else{
        finalLearningRate = 0.0;
    }
    iterations = Integer.parseInt(iterationsTextField.getText());

    learningRateFunction = learningRateFunctionCombo.getSelectedIndex() + 1;
    neighborhoodFunction = neighborhoodFunctionCombo.getSelectedIndex() + 1;
    nodeInfluenceFunction = nodeInfluenceFunctionCombo.getSelectedIndex() + 1;

    if(mapWidth < 0 || mapHeight < 0 || initialLearningRate < 0 ||
        initialLearningRate > 1 || finalLearningRate < 0 ||
            finalLearningRate > 1 || iterations < 0){
        cl.throwErrorMessage("Invalid Input! \n");
        return;
    }
}
catch(NumberFormatException e){
    //Incomplete Input
    cl.throwErrorMessage("Error on Input Parameters. \n");
    cl.throwNormalMessage("Please make sure that all fields have numeric values.
        \n");
    return;
}

int choice = 0;
if(trainer != null){
    Object[] options = {"Yes", "No"};
    choice = JOptionPane.showOptionDialog(this,
        "Are you sure you want to retrain the map?",
        "Warning",
        JOptionPane.YES_NO_OPTION,
        JOptionPane.WARNING_MESSAGE,
        null,
        options,
        options[1]);

}

if(choice == 0){
    trainer = null;
    comp = null;
    clust = null;
    umat = null;
    inputDataClass = null;
    resultPanel.removeAll();
```

```
uMatrixVisualizeButton.setEnabled(false);
clusterVisualizeButton.setEnabled(false);
componentPlaneVisualizeButton.setEnabled(false);
gisVisualizeButton.setEnabled(false);
if(!inputFileTextField.getText().equals("")){
    excelFile = new File(inputFileTextField.getText());
    if(excelFile.getAbsolutePath().matches(".*\\.xls")){
        if(excelFile.exists()){
            try {
                Workbook dataWorkbook = Workbook.getWorkbook(excelFile);

                //get contents of the workbook and store it in an array
                Sheet dataSheet = dataWorkbook.getSheet(0);
                double[][] inputData = new double[dataSheet.getRows()-1][
                    dataSheet.getColumns()-1];
                String[] nodeLabels = new String[dataSheet.getRows()-1];
                String[] variableHeaders = new String[dataSheet.getColumns()
                    -1];

                for(int i=0; i<dataSheet.getRows(); i++){

                    if(i==0){
                        Cell[] variableNames = dataSheet.getRow(0);
                        for(int j=1; j<dataSheet.getColumns(); j++){
                            variableHeaders[j-1] = variableNames[j].
                                getContents();

                        }
                    }
                    else{
                        Cell[] content = dataSheet.getRow(i);
                        nodeLabels[i-1] = content[0].getContents();
                        for(int j=1; j<dataSheet.getColumns(); j++){
                            inputData[i-1][j-1] = Double.parseDouble(content[
                                j].getContents());
                        }
                    }
                }

                maxMin = new double[inputData[0].length][2]; //0 for max, 1
                    for min

                for(int j=0; j<inputData[0].length; j++){
                    double minValue = inputData[0][j];
                    double maxValue = inputData[0][j];
                    for(int i=1; i<inputData.length; i++){
                        if (inputData[i][j] < minValue){
                            minValue = inputData[i][j];
                        }
                        if (inputData[i][j] > maxValue){
                            maxValue = inputData[i][j];
                        }
                    }

                    maxMin[j][0] = maxValue;
                    maxMin[j][1] = minValue;

                    for(int i=0; i<inputData.length; i++){
                        inputData[i][j] = (inputData[i][j] - minValue)/(
                            maxValue-minValue);
                    }
                }

                inputDataClass = new InputData(inputData.length, inputData
                    [0].length);
                inputDataClass.setDataLabels(nodeLabels);
                inputDataClass.setInputData(inputData);
                inputDataClass.setVariableLabels(variableHeaders);
```

```
            }
            catch (IOException ex) {
                cl.throwErrorMessage("Error in File Input! \n");
                return;

            }
            catch (BiffException ex) {
                //Logger.getLogger(SOMVisualizeUI.class.getName()).log(Level.
                    SEVERE, null, ex);
                cl.throwErrorMessage("File format not supported. Please check
                     input file. \n");
                cl.throwNormalMessage("Files should be in .xls or .csv format
                    . \n");
                return;
            }
        }
        else{
            cl.throwErrorMessage("File does not exist!\n");
            return;
        }
    }
    else if(excelFile.getAbsolutePath().contains(".csv")){

        if(excelFile.exists()){
            try {
                BufferedReader bufRdr = new BufferedReader(new FileReader(
                    excelFile));

                //count number of rows and cols first
                String line = null;
                int row = 0;
                int col = 0;
                while((line = bufRdr.readLine()) != null){
                    StringTokenizer st = new StringTokenizer(line, ",");

                    if(row == 0){
                        col = st.countTokens();
                    }

                    row++;
                }

                //now parse the file
                bufRdr.close();
                bufRdr = new BufferedReader(new FileReader(excelFile));

                String[] variableHeaders = new String[col];
                String[] nodeLabels = new String[row-1];
                double[][] inputData = new double[row-1][col];

                //read Variable Headers

                line = bufRdr.readLine();
                StringTokenizer st = new StringTokenizer(line, ",");

                int j=0;
                while(st.hasMoreTokens()){
                    variableHeaders[j] = st.nextToken();
                    j++;
                }

                for(int i=0; i<row-1; i++){
                    line = bufRdr.readLine();
                    st = new StringTokenizer(line, ",");

                    nodeLabels[i] = st.nextToken();
```

```
                    for (j=0; j<col; j++){
                        inputData[i][j] = Double.parseDouble(st.nextToken());
                    }
                }

                maxMin = new double[inputData[0].length][2]; //0 for max, 1
                    for min

                for(j=0; j<inputData[0].length; j++){
                    double minValue = inputData[0][j];
                    double maxValue = inputData[0][j];
                    for(int i=1; i<inputData.length; i++){
                        if (inputData[i][j] < minValue){
                            minValue = inputData[i][j];
                        }
                        if (inputData[i][j] > maxValue){
                            maxValue = inputData[i][j];
                        }
                    }

                    maxMin[j][0] = maxValue;
                    maxMin[j][1] = minValue;

                    for(int i=0; i<inputData.length; i++){
                        inputData[i][j] = (inputData[i][j] - minValue)/(
                            maxValue-minValue);
                    }
                }
                inputDataClass = new InputData(inputData.length, inputData
                    [0].length);
                inputDataClass.setDataLabels(nodeLabels);
                inputDataClass.setInputData(inputData);
                inputDataClass.setVariableLabels(variableHeaders);

            }
            catch(IOException ex){
                cl.throwErrorMessage("Error in File Input!\n");
                return;
            }
        }
        else{
            cl.throwErrorMessage("File does not exist!\n");
            return;
        }
    }
    else{
        cl.throwErrorMessage("File format not supported. Please check input
            file. \n");
        cl.throwNormalMessage("Files should be in .xls or .csv format. \n");
        return;
    }
}
else if(inputFac != null){
    if(inputFac.isReady()){

        DefaultTableModel model = inputFac.getModel();
        double[][] inputData = new double[model.getRowCount()][model.
            getColumnCount()-1];
        String[] variableHeaders = new String[model.getColumnCount()-1];
        String[] nodeLabels = new String[model.getRowCount()];

        try{
            for(int i=0; i<model.getRowCount(); i++){
                for(int j=0; j<model.getColumnCount()-1; j++){
                    inputData[i][j] = Double.parseDouble(model.getValueAt(i,
                        j+1).toString());
                }
            }
```

```
                    for(int i=0; i<model.getRowCount(); i++){
                        nodeLabels[i] = model.getValueAt(i, 0).toString();
                    }

                    for(int j=0; j<model.getColumnCount()-1; j++){
                        variableHeaders[j] = model.getColumnName(j+1);
                    }
                    maxMin = new double[inputData[0].length][2]; //0 for max, 1 for
                        min
                        for(int j=0; j<inputData[0].length; j++){
                            double minValue = inputData[0][j];
                            double maxValue = inputData[0][j];

                            for(int i=1; i<inputData.length; i++){

                                if (inputData[i][j] < minValue){
                                    minValue = inputData[i][j];
                                }
                                if (inputData[i][j] > maxValue){
                                    maxValue = inputData[i][j];
                                }
                            }



                            maxMin[j][0] = maxValue;
                            maxMin[j][1] = minValue;

                            for(int i=0; i<inputData.length; i++){
                                inputData[i][j] = (inputData[i][j] - minValue)/(
                                    maxValue-minValue);
                                if(Double.isNaN(inputData[i][j])){
                                    inputData[i][j] = 0;

                                }
                            }
                        }
                        inputDataClass = new InputData(inputData.length, inputData
                            [0].length);
                        inputDataClass.setDataLabels(nodeLabels);
                        inputDataClass.setInputData(inputData);
                        inputDataClass.setVariableLabels(variableHeaders);
                }
                catch(NumberFormatException e){
                    cl.throwErrorMessage("Error in data input format! \n");
                    return;
                }
                catch(NullPointerException e){
                    cl.throwErrorMessage("Blank input in data! \n");
                    return;
                }
            }
            else{
                cl.throwErrorMessage("Data has not been submitted! \n");
            }
        }
        else{
            cl.throwErrorMessage("No Input Data Found! \n");
            return;
        }
    }

    if(inputDataClass != null && choice == 0){
        /*
         * Train data!
         * Pass arguments INPUTDATA, MAPWIDTH, MAPHEIGHT, INITIALLEARNINGRATE,
         * FINALLEARNINGRATE, ITERATIONS, LRFUNCTION, NEIGHBORHOODFUNCTION,
             NODEINFLUENCE
```

```
         */
        cl.throwNormalMessage("Training Map. Please wait. \n");
        if(trainer == null){
            //there is no existing trainer. This may be caused by a new instance of
                the program.
            //or the trainer might have been stopped
            trainer = new SOMTrainer(inputDataClass, mapWidth, mapHeight,
                initialLearningRate, finalLearningRate, iterations,
                learningRateFunction,
                    neighborhoodFunction, nodeInfluenceFunction, cl);
        }

        mapTrainer = new Thread(trainer);
        mapTrainer.start();
        new Thread(new Enabler()).start();

    }
}//GEN-LAST:event_trainButtonActionPerformed

private void uMatrixVisualizeButtonActionPerformed(java.awt.event.ActionEvent evt) {
    //GEN-FIRST:event_uMatrixVisualizeButtonActionPerformed
    if(Integer.parseInt(neighborhoodSizeTextField.getText()) < 0){
        cl.throwErrorMessage("Invalid Input! \n");
        return;
    }

    cl.throwNormalMessage("Visualizing U-Matrix. \n");
    for(int i=0; i<resultPanel.getTabCount(); i++){
        if(resultPanel.getTitleAt(i).equals("U-Matrix")){
            resultPanel.remove(i);
        }
    }

   if(neighborhoodSizeTextField.getText().equals("")){
       cl.throwErrorMessage("No Neighborhood Size Specified! \n");
   }
   else{
       try{
           umatVisualize();
           cl.throwSuccessMessage("U-Matrix Visualization Completed! \n");
           umatDone = true;
       }
       catch(NumberFormatException e){
           cl.throwErrorMessage("Invalid input for Neighborhood Size! \n");
       }
   }
}//GEN-LAST:event_uMatrixVisualizeButtonActionPerformed

private void clusterVisualizeButtonActionPerformed(java.awt.event.ActionEvent evt) {
    //GEN-FIRST:event_clusterVisualizeButtonActionPerformed
    if(Integer.parseInt(numberOfClustersTextField.getText()) < 0){
        cl.throwErrorMessage("Invalid Input! \n");
        return;
    }

    cl.throwNormalMessage("Visualizing Clusters. \n");
    for(int i=0; i<resultPanel.getTabCount(); i++){
        if(resultPanel.getTitleAt(i).equals("Clusters")){
            resultPanel.remove(i);
        }
    }

    if(numberOfClustersTextField.getText().equals("")){
        cl.throwErrorMessage("No Cluster Count Specified! \n");
    }
    else if(iterationsClustersTextField.getText().equals("")){
        cl.throwErrorMessage("No iterations Specified! \n");
    }
```

```
        else if(Integer.parseInt(iterationsClustersTextField.getText()) < 0 || Integer.
            parseInt(iterationsClustersTextField.getText()) > 100){
            cl.throwErrorMessage("Error in iteration count! \n");
        }
        else{
            try{
                clustVisualize();
                cl.throwSuccessMessage("Cluster Visualization Completed! \n");

            }
            catch(NumberFormatException e){
                cl.throwErrorMessage("Invalid input for Number of Clusters! \n");
            }

        }
    }//GEN-LAST:event_clusterVisualizeButtonActionPerformed

    private void componentPlaneVisualizeButtonActionPerformed(java.awt.event.ActionEvent
        evt) {//GEN-FIRST:event_componentPlaneVisualizeButtonActionPerformed

        cl.throwNormalMessage("Visualizing Component Planes. \n");
        for(int i=0; i<resultPanel.getTabCount(); i++){
            if(resultPanel.getTitleAt(i).equals("Component Planes")){
                resultPanel.remove(i);
            }
        }
        compVisualize();
        cl.throwSuccessMessage("Component Plane Visualization Completed! \n");
        compDone = true;


    }//GEN-LAST:event_componentPlaneVisualizeButtonActionPerformed

    private void runButtonActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:
        event_runButtonActionPerformed
        if(trainer == null || inputDataClass == null){
            trainButton.doClick();
        }
        else{
            cl.throwNormalMessage("Resuming Training! \n");
            mapTrainer = new Thread(trainer);
            mapTrainer.start();

            new Thread(new Enabler()).start();
        }
    }//GEN-LAST:event_runButtonActionPerformed

    private void browseToggleButtonActionPerformed(java.awt.event.ActionEvent evt) {//GEN
        -FIRST:event_browseToggleButtonActionPerformed
    // TODO add your handling code here:
       javax.swing.JFileChooser fc = new javax.swing.JFileChooser();
       fc.setName("SOM Visualize");

       int returnVal = fc.showOpenDialog(null);

       if(returnVal == JFileChooser.APPROVE_OPTION){
            inputFileTextField.setText(fc.getSelectedFile().getAbsolutePath());
            if(!inputFileTextField.getText().equals("")){
                if(fc.getSelectedFile().getAbsolutePath().matches(".*\\.xls")){
                    if(fc.getSelectedFile().exists()){
                            try {
                                Workbook dataWorkbook = Workbook.getWorkbook(fc.
                                    getSelectedFile());
                                //get contents of the workbook and store it in an array
                                Sheet dataSheet = dataWorkbook.getSheet(0);
                                int dimension = (dataSheet.getColumns()-1) * 10;
                                int size = (int)Math.ceil(Math.sqrt(dimension));
                                mapWidthTextField.setText(Integer.toString(size));
```

```
                                mapHeightTextField.setText(Integer.toString(size));
                                iterationsTextField.setText(Integer.toString((int)Math.
                                    pow(size,2) * 500));
                            } catch (IOException ex) {
                                Logger.getLogger(SOMVisualizeUI.class.getName()).log(
                                    Level.SEVERE, null, ex);
                            } catch (BiffException ex) {
                                Logger.getLogger(SOMVisualizeUI.class.getName()).log(
                                    Level.SEVERE, null, ex);
                            }
                        }
                    }
                }
            }
        }//GEN-LAST:event_browseToggleButtonActionPerformed

        private void gisVisualizeButtonActionPerformed(java.awt.event.ActionEvent evt) {//GEN
            -FIRST:event_gisVisualizeButtonActionPerformed

            cl.throwNormalMessage("Generating xml file..\n");
            if(gis == null){
                gis = new MapBrowser("");
            }
            boolean gisFlag = gis.initializeXML(clust[Integer.parseInt(clustersIdTextField.
                getText())], comp, inputDataClass);

            if(gisFlag){
                cl.throwSuccessMessage("\"temp.xml\" successfully generated!");
                gis.runMap();
            }
        }//GEN-LAST:event_gisVisualizeButtonActionPerformed

        private void openFileButtonActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
            FIRST:event_openFileButtonActionPerformed
            //always ensure that there is no current file running else, ask the user if he
                wants to terminate
            int choice = 0;

            if(visFileExists){
                choice = 1;
                Object[] options = {"Yes", "No"};
                choice = JOptionPane.showOptionDialog(this,
                    "There is a file currently running. Are you sure you want to open file?",
                    "Warning",
                    JOptionPane.YES_NO_OPTION,
                    JOptionPane.WARNING_MESSAGE,
                    null,
                    options,
                    options[1]);
            }
            if(choice == 0){
                javax.swing.JFileChooser fc = new javax.swing.JFileChooser();

                int returnVal = fc.showOpenDialog(null);

                if(returnVal == JFileChooser.APPROVE_OPTION){
                   File openFile = fc.getSelectedFile();

                   if(openFile.getAbsolutePath().contains(".vis")){
                        try {
                            ObjectInputStream out = new ObjectInputStream(new FileInputStream
                                (openFile.getAbsolutePath()));

                            try {
                                //This would be the save file
                                saveFile = openFile;
                                //Get values first
                                String inputFile = out.readObject().toString();
```

```
String mapWidth = out.readObject().toString();
String mapHeight = out.readObject().toString();
String initialLearningRate = out.readObject().toString();
String finalLearningRate = out.readObject().toString();
String iterations = out.readObject().toString();
int learningRateFunction = (Integer)out.readObject();
int neighborhoodFunction = (Integer)out.readObject();
int nodeInfluenceFunction = (Integer)out.readObject();

String neighborhoodSize = out.readObject().toString();
String numberOfClusters = out.readObject().toString();

trainer = (SOMTrainer)out.readObject();
clust = (Cluster[])out.readObject();

inputDataClass = (InputData)out.readObject();
maxMin = (double[][])out.readObject();

umatDone = (Boolean)out.readObject();

compDone = (Boolean)out.readObject();

out.close();

inputPanel.setVisible(true);
consolePanel.setVisible(true);
resultPanel.setVisible(true);
resultPanel.setVisible(false);


//Reset Log
resultPanel.removeAll();
log.setText("");

doc = new DefaultStyledDocument();
cl = new ConsoleLog(log,doc);

inputFileTextField.setText(inputFile);
mapWidthTextField.setText(mapWidth);
mapHeightTextField.setText(mapHeight);
initialLearningRateTextField.setText(initialLearningRate);
finalLearningRateTextField.setText(finalLearningRate);
iterationsTextField.setText(iterations);
learningRateFunctionCombo.setSelectedIndex(
    learningRateFunction);
neighborhoodFunctionCombo.setSelectedIndex(
    neighborhoodFunction);
nodeInfluenceFunctionCombo.setSelectedIndex(
    nodeInfluenceFunction);

neighborhoodSizeTextField.setText(neighborhoodSize);
numberOfClustersTextField.setText(numberOfClusters);


if(trainer != null){
    trainer.setCl(cl);
    if(trainer.isDoneTraining()){
        LatticePainter lp = new LatticePainter(trainer.
            getLattice(), inputDataClass);
        lp.setBounds(0,0,450,480);
        resultPanel.add("SOM Lattice",lp);

        uMatrixVisualizeButton.setEnabled(true);
        clusterVisualizeButton.setEnabled(true);
        componentPlaneVisualizeButton.setEnabled(true);

        umatButton.setEnabled(true);
        clustButton.setEnabled(true);
```

```
                                compButton.setEnabled(true);
                                resultPanel.setVisible(true);
                            }
                        }

                        if(clust != null){
                            clustVisualize();
                        }

                        if(umatDone){
                            umatVisualize();
                        }
                        if(compDone){
                            compVisualize();
                        }
                        if(resultPanel.getTabCount() > 0){
                            resultPanel.setVisible(true);
                        }

                        this.visFileExists = true;
                        this.setTitle("SOMVisualize - " + openFile.getName());

                        runButton.setEnabled(true);
                        pauseButton.setEnabled(true);
                        stopButton.setEnabled(true);
                        cl.throwSuccessMessage("File has been successfully loaded.\n"
                            );


                    } catch (ClassNotFoundException ex) {
                        cl.throwErrorMessage("File is corrupted! \n");
                    } catch (NumberFormatException ex){
                        cl.throwErrorMessage("File may not have loaded successfully!
                            \nSome parts may have been corrupted. \n");
                    }

                } catch (IOException ex) {
                    JOptionPane.showMessageDialog(this, "Error in opening file! \n",
                        "SOM Visualize - Fatal Error", JOptionPane.ERROR_MESSAGE);
                    saveFile = null;
                }
            }
        }
    }
}//GEN-LAST:event_openFileButtonActionPerformed

private void pauseButtonActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:
    event_pauseButtonActionPerformed
    cl.throwErrorMessage("Training has been paused! \n");
    cl.throwNormalMessage("You may resume training by pressing the play button. \n");
    trainer.pause();
}//GEN-LAST:event_pauseButtonActionPerformed

private void inputFacilityToggleButtonActionPerformed(java.awt.event.ActionEvent evt)
     {//GEN-FIRST:event_inputFacilityToggleButtonActionPerformed
    //create new instance of input facility!
    if(inputFac == null){
        inputFac = new InputFacility();
    }
    else{
        inputFac.setVisible(true);
    }
}//GEN-LAST:event_inputFacilityToggleButtonActionPerformed

private void neighborhoodSizeTextFieldCaretUpdate(javax.swing.event.CaretEvent evt) {
    //GEN-FIRST:event_neighborhoodSizeTextFieldCaretUpdate
    try{
        int temp = Integer.parseInt(neighborhoodSizeTextField.getText());
```

```
            neighborhoodSizeTextField.setBackground(new Color(0,255,200));

            if(temp < 0){
                neighborhoodSizeTextField.setBackground(new Color(255,200,200));
            }
        }
        catch(NumberFormatException e){
            neighborhoodSizeTextField.setBackground(new Color(255,200,200));
        }
    }//GEN-LAST:event_neighborhoodSizeTextFieldCaretUpdate

    private void mapWidthTextFieldCaretUpdate(javax.swing.event.CaretEvent evt) {//GEN-
        FIRST:event_mapWidthTextFieldCaretUpdate
        try{
            int temp = Integer.parseInt(mapWidthTextField.getText());
            if(temp < 1){
                mapWidthTextField.setBackground(new Color(255,200,200));
            }
            else if(temp < 51){
                mapWidthTextField.setBackground(new Color(0,255,200));
            }
            else{
                mapWidthTextField.setBackground(new Color(255,200,200));
            }
        }
        catch(NumberFormatException e){
            mapWidthTextField.setBackground(new Color(255,200,200));
        }
    }//GEN-LAST:event_mapWidthTextFieldCaretUpdate

    private void mapHeightTextFieldCaretUpdate(javax.swing.event.CaretEvent evt) {//GEN-
        FIRST:event_mapHeightTextFieldCaretUpdate
        try{
            int temp = Integer.parseInt(mapHeightTextField.getText());
            if(temp < 1){
                mapHeightTextField.setBackground(new Color(255,200,200));
            }
            else if(temp < 51){
                mapHeightTextField.setBackground(new Color(0,255,200));
            }
            else{
                mapHeightTextField.setBackground(new Color(255,200,200));
            }
        }
        catch(NumberFormatException e){
            mapHeightTextField.setBackground(new Color(255,200,200));
        }
    }//GEN-LAST:event_mapHeightTextFieldCaretUpdate

    private void initialLearningRateTextFieldCaretUpdate(javax.swing.event.CaretEvent evt
        ) {//GEN-FIRST:event_initialLearningRateTextFieldCaretUpdate
        try{
            double temp = Double.parseDouble(initialLearningRateTextField.getText());
            if(temp < 0){
                initialLearningRateTextField.setBackground(new Color(255,200,200));
            }
            else if(temp <= 1){
                initialLearningRateTextField.setBackground(new Color(0,255,200));
            }
            else{
                initialLearningRateTextField.setBackground(new Color(255,200,200));
            }
        }
        catch(NumberFormatException e){
            initialLearningRateTextField.setBackground(new Color(255,200,200));
        }
    }//GEN-LAST:event_initialLearningRateTextFieldCaretUpdate
```

```
private void finalLearningRateTextFieldCaretUpdate(javax.swing.event.CaretEvent evt)
    {//GEN-FIRST:event_finalLearningRateTextFieldCaretUpdate
    try{
        double temp = Double.parseDouble(finalLearningRateTextField.getText());
        if(temp < 0){
            finalLearningRateTextField.setBackground(new Color(255,200,200));
        }
        else if(temp <= 1){
            finalLearningRateTextField.setBackground(new Color(0,255,200));
        }
        else{
            finalLearningRateTextField.setBackground(new Color(255,200,200));
        }
    }
    catch(NumberFormatException e){
        finalLearningRateTextField.setBackground(new Color(255,200,200));
    }
}//GEN-LAST:event_finalLearningRateTextFieldCaretUpdate

private void iterationsTextFieldCaretUpdate(javax.swing.event.CaretEvent evt) {//GEN-
    FIRST:event_iterationsTextFieldCaretUpdate
    try{
        int temp = Integer.parseInt(iterationsTextField.getText());
        if(temp < 0){
            iterationsTextField.setBackground(new Color(255,200,200));
        }
        else if(temp <= 1250000){
            iterationsTextField.setBackground(new Color(0,255,200));
        }
        else{
            iterationsTextField.setBackground(new Color(255,200,200));
        }
    }
    catch(NumberFormatException e){
        iterationsTextField.setBackground(new Color(255,200,200));
    }
}//GEN-LAST:event_iterationsTextFieldCaretUpdate

private void numberOfClustersTextFieldCaretUpdate(javax.swing.event.CaretEvent evt) {
    //GEN-FIRST:event_numberOfClustersTextFieldCaretUpdate
    try{
        int temp = Integer.parseInt(numberOfClustersTextField.getText());
        numberOfClustersTextField.setBackground(new Color(0,255,200));

        if(temp < 0){
            numberOfClustersTextField.setBackground(new Color(255,200,200));
            trainButton.setEnabled(false);
        }
    }
    catch(NumberFormatException e){
        numberOfClustersTextField.setBackground(new Color(255,200,200));
    }
}//GEN-LAST:event_numberOfClustersTextFieldCaretUpdate

private void jMenuItem1ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:
    event_jMenuItem1ActionPerformed
        File tutorial = new File("Tutorial/tutorial.html");

        try {
            Runtime.getRuntime().exec("rundll32 url.dll,FileProtocolHandler " +
                tutorial.toURI());
        } catch (IOException ex) {
            Logger.getLogger(SOMVisualizeUI.class.getName()).log(Level.SEVERE, null,
                ex);
        }


}//GEN-LAST:event_jMenuItem1ActionPerformed
```

```
private void umatButtonActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:
    event_umatButtonActionPerformed
    uMatrixVisualizeButton.doClick();
}//GEN-LAST:event_umatButtonActionPerformed

private void clustButtonActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:
    event_clustButtonActionPerformed
    clusterVisualizeButton.doClick();
}//GEN-LAST:event_clustButtonActionPerformed

private void compButtonActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:
    event_compButtonActionPerformed
    componentPlaneVisualizeButton.doClick();
}//GEN-LAST:event_compButtonActionPerformed

private void saveAsFileMenuActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
    FIRST:event_saveAsFileMenuActionPerformed
    saveFile = null;
    saveFileButton.doClick();
}//GEN-LAST:event_saveAsFileMenuActionPerformed

private void openFileMenuActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST
    :event_openFileMenuActionPerformed
    openFileButton.doClick();
}//GEN-LAST:event_openFileMenuActionPerformed

private void iterationsClustersTextFieldCaretUpdate(javax.swing.event.CaretEvent evt)
    {//GEN-FIRST:event_iterationsClustersTextFieldCaretUpdate
try{
        int temp = Integer.parseInt(iterationsClustersTextField.getText());
        if(temp < 0){
            iterationsClustersTextField.setBackground(new Color(255,200,200));
        }
        else if(temp <= 100){
            iterationsClustersTextField.setBackground(new Color(0,255,200));
        }
        else{
            iterationsClustersTextField.setBackground(new Color(255,200,200));
        }
    }
    catch(NumberFormatException e){
        iterationsClustersTextField.setBackground(new Color(255,200,200));
    }
}//GEN-LAST:event_iterationsClustersTextFieldCaretUpdate

private void clustersIdTextFieldCaretUpdate(javax.swing.event.CaretEvent evt) {//GEN-
    FIRST:event_clustersIdTextFieldCaretUpdate
// TODO add your handling code here:
}//GEN-LAST:event_clustersIdTextFieldCaretUpdate

private void clustersIdTextFieldActionPerformed(java.awt.event.ActionEvent evt) {//
    GEN-FIRST:event_clustersIdTextFieldActionPerformed
// TODO add your handling code here:
}//GEN-LAST:event_clustersIdTextFieldActionPerformed

public void umatVisualize(){
    umat =  new Umatrix(trainer.getLattice(),Integer.parseInt(
        neighborhoodSizeTextField.getText()), inputDataClass);
    umat.run();

    double maxAverage = 0;
    double minAverage = 0;

    for(int i=0; i<maxMin.length; i++){
       maxAverage = maxAverage+maxMin[i][0];
       minAverage = minAverage+maxMin[i][1];
    }
```

```
        //take average
        maxAverage = maxAverage/maxMin.length;
        minAverage = minAverage/maxMin.length;

        //adjust value

        maxAverage = (umat.getMaxDistance())*(maxAverage-minAverage)+ minAverage;
        minAverage = (umat.getMinDistance())*(maxAverage-minAverage)+ minAverage;

        umat.setOrigMaxMin(maxAverage, minAverage);
        umat.setBounds(0,0,450,480);


        resultPanel.add("U-Matrix", umat);
        resultPanel.setSelectedIndex(resultPanel.getTabCount()-1);


    }

    public void clustVisualize(){
        if(clust == null){
            clust = new Cluster[Integer.parseInt(iterationsClustersTextField.getText())];
        }
        JPanel pan = new javax.swing.JPanel();
        pan.setLayout(null);

        for(int i=0; i<clust.length; i++){
            if(clust[i] == null){
                clust[i] = new Cluster(trainer.getLattice(), Integer.parseInt(
                    numberOfClustersTextField.getText()));
                clust[i].run();
            }
            LatticePainter lp = new LatticePainter(clust[i].getLattice(), trainer.
                getInputData(), 3);
            lp.setMaxMin(clust[i].getNumberOfClusters(), 0);
            lp.setBounds((240-225)/2, (280-240)/2, 225, 240);
            JPanel lab = new JPanel();
            lab.setLayout(null);
            lab.add(lp);
            lab.setBorder(javax.swing.BorderFactory.createTitledBorder("Cluster Run " + i
                ));
            lab.setBounds(240*(i%4),(280*(i/4)),240, 280);
            lab.setToolTipText("Cluster Run" + i);
            pan.add(lab);

        }
        pan.setPreferredSize(new java.awt.Dimension(240*4, 280*(clust.length/4+1)));
        javax.swing.JScrollPane scrollPan = new javax.swing.JScrollPane(pan);
        resultPanel.add("Clusters", scrollPan);
        resultPanel.setSelectedIndex(resultPanel.getTabCount()-1);

        if(comp != null){
            gisVisualizeButton.setEnabled(true);
        }
    }

    public void compVisualize(){

        comp = new Component[trainer.getLattice().getNumberOfNodeElements()];

        JPanel pan = new javax.swing.JPanel();
        pan.setLayout(null);

        for(int i=0; i<trainer.getLattice().getNumberOfNodeElements(); i++){
            comp[i] = new Component(trainer.getLattice(), i);
            comp[i].setBounds((240-225)/2, (280-240)/2, 225, 240);
            comp[i].setOrigMaxMin(maxMin[i][0], maxMin[i][1]);
```

```
            JPanel lab = new JPanel();
            lab.setLayout(null);
            lab.add(comp[i]);
            lab.setBorder(javax.swing.BorderFactory.createTitledBorder(inputDataClass.
                getVariableLabels()[i]));
            lab.setBounds(240*(i%4),(280*(i/4)),240, 280);
            lab.setToolTipText(inputDataClass.getVariableLabels()[i]);
            pan.add(lab);

        }
        pan.setPreferredSize(new java.awt.Dimension(240*4, 280*(trainer.getLattice().
            getNumberOfNodeElements()/4+1)));
        javax.swing.JScrollPane scrollPan = new javax.swing.JScrollPane(pan);
        resultPanel.add("Component Planes", scrollPan);
        resultPanel.setSelectedIndex(resultPanel.getTabCount()-1);

        if(clust != null){
            gisVisualizeButton.setEnabled(true);
        }
    }

    private void showStartUpScreen(){
        JFrame startUpScreen = new JFrame();
        JLabel startUpImage = new JLabel(new ImageIcon(getClass().getResource("/my/
            SOMVisualize/Icons/Splash Screen.png")));

        startUpScreen.setSize(507, 294);
        startUpScreen.setLocation((this.getWidth()/2) - startUpScreen.getWidth()/2, (this
            .getHeight()/2) - startUpScreen.getHeight()/2);
        startUpScreen.setTitle("SOM Visualize - Welcome");
        startUpScreen.setAlwaysOnTop(true);
        startUpScreen.add(startUpImage);
        startUpScreen.setVisible(true);
    }


    // Variables declaration - do not modify//GEN-BEGIN:variables
    private javax.swing.JButton browseToggleButton;
    private javax.swing.JButton clustButton;
    private javax.swing.JLabel clusterDetailsGroup;
    private javax.swing.JButton clusterVisualizeButton;
    private javax.swing.JLabel clustersIdLabel;
    private javax.swing.JTextField clustersIdTextField;
    private javax.swing.JButton compButton;
    private javax.swing.JLabel componentPlaneDetailsGroup;
    private javax.swing.JButton componentPlaneVisualizeButton;
    private javax.swing.JScrollPane consolePanel;
    private javax.swing.JMenu editMenu;
    private javax.swing.JMenu fileMenu;
    private javax.swing.JCheckBox finalLearningRateEnable;
    private javax.swing.JLabel finalLearningRateLabel;
    private javax.swing.JTextField finalLearningRateTextField;
    private javax.swing.JLabel gisDetailsGroup;
    private javax.swing.JButton gisVisualizeButton;
    private javax.swing.JLabel initialLearningRateLabel;
    private javax.swing.JTextField initialLearningRateTextField;
    private javax.swing.JLabel inputDataGroup;
    private javax.swing.JButton inputFacilityToggleButton;
    private javax.swing.JLabel inputFileLabel;
    private javax.swing.JTextField inputFileTextField;
    private javax.swing.JScrollPane inputPanel;
    private javax.swing.JPanel inputPanelHolder;
    private javax.swing.JSeparator inputPanelSeparator1;
    private javax.swing.JSeparator inputPanelSeparator2;
    private javax.swing.JSeparator inputPanelSeparator3;
    private javax.swing.JSeparator inputPanelSeparator4;
    private javax.swing.JSeparator inputPanelSeparator5;
    private javax.swing.JSeparator inputPanelSeparator6;
```

```
private javax.swing.JLabel iterationsClustersLabel;
private javax.swing.JTextField iterationsClustersTextField;
private javax.swing.JLabel iterationsLabel;
private javax.swing.JTextField iterationsTextField;
private javax.swing.JFileChooser jFileChooser1;
private javax.swing.JMenu jMenu1;
private javax.swing.JMenuBar jMenuBar1;
private javax.swing.JMenuItem jMenuItem1;
private javax.swing.JToolBar.Separator jSeparator1;
private javax.swing.JComboBox learningRateFunctionCombo;
private javax.swing.JLabel learningRateFunctionLabel;
private javax.swing.JTextPane log;
private javax.swing.JLabel mapHeightLabel;
private javax.swing.JTextField mapHeightTextField;
private javax.swing.JLabel mapWidthLabel;
private javax.swing.JTextField mapWidthTextField;
private javax.swing.JComboBox neighborhoodFunctionCombo;
private javax.swing.JLabel neighborhoodFunctionLabel;
private javax.swing.JLabel neighborhoodSizeLabel;
private javax.swing.JTextField neighborhoodSizeTextField;
private javax.swing.JButton newFileButton;
private javax.swing.JMenuItem newFileMenu;
private javax.swing.JComboBox nodeInfluenceFunctionCombo;
private javax.swing.JLabel nodeInfluenceFunctionLabel;
private javax.swing.JLabel numberOfClustersLabel;
private javax.swing.JTextField numberOfClustersTextField;
private javax.swing.JButton openFileButton;
private javax.swing.JMenuItem openFileMenu;
private javax.swing.JButton pauseButton;
private javax.swing.JTabbedPane resultPanel;
private javax.swing.JButton runButton;
private javax.swing.JMenuItem saveAsFileMenu;
private javax.swing.JButton saveFileButton;
private javax.swing.JMenuItem saveFileMenu;
private javax.swing.JLabel somDetailsGroup;
private javax.swing.JButton stopButton;
private javax.swing.JToolBar toolBar;
private javax.swing.JToolBar.Separator toolBarSeparator1;
private javax.swing.JButton trainButton;
private javax.swing.JLabel uMatrixDetailsGroup;
private javax.swing.JButton uMatrixVisualizeButton;
private javax.swing.JButton umatButton;
// End of variables declaration//GEN-END:variables

//File input and input data
private File excelFile;
private InputData inputDataClass;

private InputFacility inputFac;
//Trainer and visualizers
private SOMTrainer trainer;
private Umatrix umat;
private Cluster[] clust;
private Component[] comp;
private MapBrowser gis;

//Console
private StyledDocument doc;
private ConsoleLog cl;

//Miscellaneous
private double[][] maxMin;
private Thread mapTrainer;

//File Output
private File saveFile;

//Flags
```

```
    boolean visFileExists;
    boolean fileIsSaved;
    boolean umatDone;
    boolean compDone;


    class Enabler implements Runnable{

            @Override
            public void run() {
                while(mapTrainer.isAlive()){
                    if(trainer.isDoneTraining()){
                        boolean trainedFlag = false;
                        for(int i=0; i<resultPanel.getTabCount(); i++){
                            if(resultPanel.getTitleAt(i).equals("SOM Lattice")){
                                trainedFlag = true;
                            }
                        }
                        if(!trainedFlag){
                            LatticePainter lp = new LatticePainter(trainer.getLattice
                                (), inputDataClass);
                            lp.setBounds(0,0,450,480);
                            resultPanel.add("SOM Lattice",lp);
                        }
                        uMatrixVisualizeButton.setEnabled(true);
                        clusterVisualizeButton.setEnabled(true);
                        componentPlaneVisualizeButton.setEnabled(true);

                        umatButton.setEnabled(true);
                        clustButton.setEnabled(true);
                        compButton.setEnabled(true);

                        resultPanel.setVisible(true);
                        break;
                    }
                    else{
                    }
                }
            }
        }
}
```

SOMVisualize/SOMTrainer.java

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package my.SOMVisualize;

import java.awt.*;
import java.io.Serializable;
import java.util.logging.Level;
import java.util.logging.Logger;

/**
 *
 * @author Mark Lester
 */
public class SOMTrainer implements Runnable, Serializable{

    private InputData inputData;

    private int mapWidth;
    private int mapHeight;

    private double initialLearningRate;
    private double finalLearningRate;
```

```
private int currIter = 0;
private int iterations;

private int prevValue = 0;


private int learningRateFunction; // 1 for exponential, 2 for power series, 3 for
     custom
private int neighborhoodFunction; // 1 for exponential, 2 for custom
private int nodeInfluenceFunction; // 1 for Gaussian, 2 for custom

private boolean doneTraining = false;
private boolean stopTraining = false;
private boolean pauseTraining = false;

private SOMLattice lattice;

private ConsoleLog cl;

public SOMTrainer(InputData inputData, int mapWidth, int mapHeight, double
    initialLearningRate, double finalLearningRate, int iterations,
                    int learningRateFunction, int neighborhoodFunction, int
                        nodeInfluenceFunction, ConsoleLog cl){

    this.inputData = inputData;
    this.mapWidth = mapWidth;
    this.mapHeight = mapHeight;
    this.initialLearningRate = initialLearningRate;
    this.finalLearningRate = finalLearningRate;
    this.iterations = iterations;

    this.learningRateFunction = learningRateFunction;
    this.neighborhoodFunction = neighborhoodFunction;
    this.nodeInfluenceFunction = nodeInfluenceFunction;
    this.cl = cl;
}

public SOMTrainer(InputData inputData, ConsoleLog cl){
    this(inputData,40,40,0.7,0.001,500,1,1,1, cl);
}

private boolean train(){
    if(doneTraining){   //do not allow
        return true;    //retraining
    }                   //of the map
    /*
     * Initialize the map and the nodes
     */

    if(currIter == 0){
        lattice = new SOMLattice(this.mapWidth, this.mapHeight, inputData.getCols
            ()); /* original implementation*/
    }
    while(currIter < iterations){
        /*
         * If stop button is pressed. Stop the thread and training!
         */
        if(stopTraining || pauseTraining){
            break;
        }
        /*
         * Choose an input vector randomly from the set.
         */

        int timesHundred = currIter*100;
        if(prevValue == (timesHundred/iterations)-10){
```

```
                System.out.println(prevValue);
                cl.throwNormalMessage("Training in progress: " + timesHundred/
                    iterations + "% completed\n");
                prevValue = timesHundred/iterations;
            }

            int inputDataIndex = (int)Math.floor(Math.random()*(inputData.getRows()))
                ;
            /*
             * Find best matching unit of the chosen input vector
             */
            Node winningNode = lattice.getBMU(inputData.getInputData()[inputDataIndex
                ]);
            inputData.setOneBMU(inputDataIndex, winningNode.getNodeIndex());
             /*
             * Calculate the width of the neighborhood at this timestep
             */
            double neighborhoodRadius = 0;

            if(neighborhoodFunction == 1){
                neighborhoodRadius = lattice.getExponentialNeighborhood(currIter,
                    iterations);
            }
            else{
            }

            /*
             * Compute for the current learning rate.
             */
            double currLearningRate = 0;
            if(learningRateFunction==1){
                currLearningRate = lattice.getExponentialLearning(initialLearningRate
                    , currIter, iterations);
            }
            else if(learningRateFunction==2){
                currLearningRate = lattice.getPowerSeriesLearning(initialLearningRate
                    , finalLearningRate, currIter, iterations);
            }
            else{
            }

            /*
             * Adjust the lattice.
             */
            lattice.adjustNeighborhood(winningNode, neighborhoodRadius,
                currLearningRate, inputData.getInputData()[inputDataIndex],
                nodeInfluenceFunction);

            currIter++;
        }

        if(!stopTraining && !pauseTraining){
            doneTraining = true;
            cl.throwSuccessMessage("Training Completed! \n");
            cl.throwNormalMessage("Enabling Visualization Buttons. \n");
        }
        return false;
    }

    public SOMLattice getLattice() {
        return lattice;
    }

    public InputData getInputData() {
        return inputData;
    }

    public boolean isDoneTraining() {
```

```
        return doneTraining;
    }

    @Override
    public void run() {
        stopTraining = false;
        pauseTraining = false;
        this.train();
    }

    public void stop(){
        stopTraining = true;
    }

    public void pause(){
        pauseTraining = true;
    }

    public void setCl(ConsoleLog cl) {
        this.cl = cl;
    }
}
```

SOMVisualize/SOMLattice.java

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package my.SOMVisualize;

import java.awt.Color;
import java.io.Serializable;

/**
 *
 * @author Mark Lester
 */
public class SOMLattice implements Serializable{

    private int latticeWidth;
    private int latticeHeight;

    private int numberOfNodeElements; // number of elements inside a Node

    private int nodeWidth;
    private int nodeHeight;
    private int totalNumberOfNodes;

    private Node[] latticeNode;

    private final int MAP_RADIUS = 225;

    public SOMLattice(int latticeWidth, int latticeHeight, int numberOfNodeElements){

        this.latticeWidth = latticeWidth;
        this.latticeHeight = latticeHeight;
        this.numberOfNodeElements = numberOfNodeElements;

        initializeLattice();
    }

    public SOMLattice(){
       this(10,10,3);
    }

    public void initializeValues(){
        totalNumberOfNodes = this.latticeWidth*this.latticeHeight;
```

```
        latticeNode = new Node[totalNumberOfNodes]; //specify the array of nodes

        nodeWidth = (int)Math.floor(450/this.latticeWidth);    //compute for the
            height and width of the node throughout
        nodeHeight = (int)Math.floor(450/this.latticeHeight); // the training. this
            should be constant.
    }
    protected void initializeLattice(){
        /*This part should create the Nodes*/
        /*Its for the lattice*/

        totalNumberOfNodes = this.latticeWidth*this.latticeHeight;

        latticeNode = new Node[totalNumberOfNodes]; //specify the array of nodes

        nodeWidth = (int)Math.floor(450/this.latticeWidth);    //compute for the
            height and width of the node throughout
        nodeHeight = (int)Math.floor(450/this.latticeHeight); // the training. this
            should be constant.



        for(int i=0; i<totalNumberOfNodes; i++){
            latticeNode[i] = new Node(((i%this.latticeWidth)*nodeWidth) + nodeWidth
                /2,((i/this.latticeWidth)*nodeHeight) + nodeHeight/2,
                numberOfNodeElements, i);
            latticeNode[i].setNodeColor(new Color((int)(latticeNode[i].
                getDoubleElementAt(0)*255),
                                                  (int)(latticeNode[i].
                                                      getDoubleElementAt(1)*255),
                                                     (int)(latticeNode[i].
                                                         getDoubleElementAt(2)*255)
                                                         ));
        }



    }



    /*public int getLatticeSize(){
        return totalNumberOfNodes;
    }*/

    public Node getBMU(double[] inputVector){

        double minDistance = 0;
        Node BMU = null;

        for(int i=0; i<this.getTotalNumberOfNodes(); i++){
            double distance = latticeNode[i].getDistance(inputVector);

            if(i==0){
                minDistance = distance;
            }

            if(distance <= minDistance){
                minDistance = distance;
                BMU = latticeNode[i];
            }
        }
        return BMU;
    }

    public double getExponentialNeighborhood(int currIter, int totalIter){
        double timeConstant = totalIter/Math.log(MAP_RADIUS);
```

```
        return (MAP_RADIUS * Math.exp(-(double)currIter/timeConstant));
    }

    public double getExponentialLearning(double initialLearningRate, int currIter,
        int totalIter){
        double timeConstant = totalIter/Math.log(MAP_RADIUS);

        return (initialLearningRate * Math.exp(-(double)currIter/timeConstant));
    }
    public double getPowerSeriesLearning(double initialLearningRate, double
        finalLearningRate, int currIter, int totalIter){
        return (initialLearningRate * Math.pow((finalLearningRate/initialLearningRate
            ), (currIter/totalIter)));
    }

    public void adjustNeighborhood(Node winningNode, double neighborhoodRadius, double
        currLearningRate, double[] inputVector,int nodeInfluenceFunction){
        for(int i=0; i<this.getTotalNumberOfNodes(); i++){
            /*
             * Calculate Euclidean Distance (squared) to this node from the BMU
             *
             */

            double distanceToNodeSquared = Math.pow((latticeNode[i].getXPos() -
                winningNode.getXPos()), 2) +
                                           Math.pow((latticeNode[i].getYPos() -
                                               winningNode.getYPos()), 2);

            double neighborhoodSquared = Math.pow(neighborhoodRadius, 2);
            double bmuInfluence = 0.000000000;

            if(distanceToNodeSquared < neighborhoodSquared){
                if(nodeInfluenceFunction == 1){
                    bmuInfluence = Math.exp(-(distanceToNodeSquared)/(2*
                        neighborhoodSquared));

                }
                else{
                }
                latticeNode[i].adjustWeights(inputVector, currLearningRate,
                    bmuInfluence);
            }
        }
    }

    public int getLatticeHeight() {
        return latticeHeight;
    }

    public void setLatticeHeight(int latticeHeight) {
        this.latticeHeight = latticeHeight;
    }

    public Node[] getLatticeNode() {
        return latticeNode;
    }

    public void setLatticeNode(Node[] latticeNode) {
        this.latticeNode = latticeNode;
    }

    public int getLatticeWidth() {
        return latticeWidth;
    }

    public void setLatticeWidth(int latticeWidth) {
        this.latticeWidth = latticeWidth;
```

```
    }

    public int getNodeHeight() {
        return nodeHeight;
    }

    public void setNodeHeight(int nodeHeight) {
        this.nodeHeight = nodeHeight;
    }

    public int getNodeWidth() {
        return nodeWidth;
    }

    public void setNodeWidth(int nodeWidth) {
        this.nodeWidth = nodeWidth;
    }

    public int getNumberOfNodeElements() {
        return numberOfNodeElements;
    }

    public void setNumberOfNodeElements(int numberOfNodeElements) {
        this.numberOfNodeElements = numberOfNodeElements;
    }

    public int getTotalNumberOfNodes() {
        return totalNumberOfNodes;
    }

    public void setTotalNumberOfNodes(int totalNumberOfNodes) {
        this.totalNumberOfNodes = totalNumberOfNodes;
    }
}
```

SOMVisualize/Node.java

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package my.SOMVisualize;

/**
 *
 * @author Mark Lester
 */

import java.awt.Color;

public class Node extends java.util.Vector{

    private int xPos; /* X and Y positions */
    private int yPos; /* of the node in the map */

    private int numberOfElements;

    private int cluster;
    private Color nodeColor; //used for visualization purposes

    private int nodeIndex;

    public Node(int xPos, int yPos, int numberOfElements, int nodeIndex){
        this.xPos = xPos;
        this.yPos = yPos;
        this.numberOfElements = numberOfElements;
        this.cluster = -1;
        this.nodeIndex = nodeIndex;
```

```
    for(int i=0; i<this.numberOfElements; i++){ //this is a dummy initializer,
        should change later on.
        this.addElement(Math.random());
    }
}

public Node(Node node){
    this.xPos = node.xPos;
    this.yPos = node.yPos;
    this.numberOfElements = node.numberOfElements;
    this.cluster = -1;
    this.nodeIndex = node.getNodeIndex();
    for(int i=0; i<this.numberOfElements; i++){ //this is a dummy initializer,
        should change later on.
        this.addElement(node.getDoubleElementAt(i));
    }
}

public int getNodeIndex() {
    return nodeIndex;
}

public int getXPos(){
    return xPos;
}

public int getYPos(){
    return yPos;
}

public void setNodeColor(Color nodeColor){
    this.nodeColor = nodeColor;
}

public Color getNodeColor(){
    return nodeColor;
}

public double getDoubleElementAt(int index){
    double element = (Double)this.elementAt(index);
    return element;
}

public void setDoubleElementAt(double value,int index){
    this.setElementAt(value, index);
}

public double getDistance(double[] inputVector){

    double distance = 0;

    for(int i=0; i<inputVector.length; i++){
       distance += Math.pow((this.getDoubleElementAt(i) - inputVector[i]), 2);

    }
    return Math.sqrt(distance);
}

public double getDistance(Node inputVector){

    double distance = 0;

    for(int i=0; i<inputVector.size(); i++){
       distance += Math.pow((this.getDoubleElementAt(i) - inputVector.
           getDoubleElementAt(i)), 2);
    }
    return Math.sqrt(distance);
```

```
    }

    public void adjustWeights(double[] inputVector, double learningRate, double
        bmuInfluence){
        for(int i=0; i<this.numberOfElements; i++){
            this.setDoubleElementAt((this.getDoubleElementAt(i) + bmuInfluence*
                learningRate*(inputVector[i] - this.getDoubleElementAt(i))), i);

            //these are partial codes
            this.setNodeColor(new Color((int)(this.getDoubleElementAt(0)*255),
                                                    (int)(this.getDoubleElementAt(1)
                                                        *255),
                                                    (int)(this.getDoubleElementAt
                                                        (2)*255)));
        }
    }

    public int getCluster() {
        return cluster;
    }

    public void setCluster(int cluster) {
        this.cluster = cluster;
    }



}
```

SOMVisualize/LatticePainter.java

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

package my.SOMVisualize;

import java.awt.image.*;
import java.awt.*;
import javax.swing.JPanel;
/**
 *
 * @author Mark Lester
 */
public class LatticePainter extends JPanel{

    private SOMLattice lattice;
    private InputData inputData;
    private int latticeType = 0;

    private int maxValue;
    private int minValue;



    public LatticePainter(SOMLattice lattice, InputData inputData){
        this.lattice = lattice;
        this.inputData = inputData;
    }

    public LatticePainter(SOMLattice lattice, InputData inputData, int latticeType){
        this.lattice = lattice;
        this.inputData = inputData;
        this.latticeType = latticeType;
    }
```

```
public void setInputData(InputData inputData) {
    this.inputData = inputData;
}

public void setLattice(SOMLattice lattice) {
    this.lattice = lattice;
}

public void setLatticeType(int latticeType){
    this.latticeType = latticeType;
}

public void setMaxMin(int max, int min){
    this.maxValue = max;
    this.minValue = min;
}


public void paintComponent(Graphics g){

    if(latticeType == 3){
        Graphics2D g2 = (Graphics2D)g;

        g2.scale(0.5, 0.5);

    }
    else{
        g.translate(this.getWidth()/2 - 225, this.getHeight()/2 - 240);
    }
    for(int i=0; i<lattice.getTotalNumberOfNodes(); i++){

        g.setColor(lattice.getLatticeNode()[i].getNodeColor());
        g.fillRect(lattice.getLatticeNode()[i].getXPos() - lattice.getNodeWidth()
            /2, lattice.getLatticeNode()[i].getYPos() - lattice.getNodeHeight()
            /2, lattice.getNodeWidth(), lattice.getNodeHeight());
        g.setColor(java.awt.Color.BLACK);
        g.drawRect(lattice.getLatticeNode()[i].getXPos() - lattice.getNodeWidth()
            /2, lattice.getLatticeNode()[i].getYPos() - lattice.getNodeHeight()
            /2, lattice.getNodeWidth(), lattice.getNodeHeight());
    }

        g.setColor(Color.BLACK);
        for(int i=0; i<inputData.getRows(); i++){
            g.drawString(inputData.getDataLabels()[i], lattice.getLatticeNode()[
                inputData.getInputDataBMUs()[i]].getXPos(), lattice.
                getLatticeNode()[inputData.getInputDataBMUs()[i]].getYPos());
        }


    if(latticeType == 3){
        for(int i=0; i<maxValue; i++){
            int color = 1024/(int)maxValue*i;
            if(color < 256){
                g.setColor(new Color(0,(color % 256),255));
            }
            else if(color < 512){
                g.setColor(new Color(0,255,255 - (color % 256)));
            }
            else if(color < 768){
                g.setColor(new Color((color % 256),255,0));
            }
            else{
                g.setColor(new Color(255,255 - (color % 256),0));
            }
            double width = 350;
            g.fillRect((int)Math.ceil(i*width/maxValue) + 50, 455, (int)Math.ceil
                (width/maxValue), 15);
```

```
            }
            g.setColor(Color.BLACK);
            g.drawRect(50,455, 350, 15);

        }


    }

}
```

SOMVisualize/Umatrix.java

```
/*
 * To change lattice template, choose Tools | Templates
 * and open the template in the editor.
 */
package my.SOMVisualize;

import java.awt.*;
import java.io.Serializable;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JPanel;

/**
 *
 * @author Mark Lester
 */
public class Umatrix extends JPanel implements Runnable, Serializable{

    private int neighborhoodSize;
    private SOMLattice lattice;
    private double maxDistance;
    private double minDistance;

    private double origMaxValue;
    private double origMinValue;

    private InputData inputData;

    public Umatrix(SOMLattice lattice, int neighborhoodSize, InputData inputData){

        this.neighborhoodSize = neighborhoodSize;
        this.lattice = new SOMLattice();
        this.inputData = inputData;
        initializeLattice(lattice);
    }

    public double getMaxDistance() {
        return maxDistance;
    }

    public double getMinDistance() {
        return minDistance;
    }

    public void setOrigMaxMin(double maxValue, double minValue){
        this.origMaxValue = maxDistance * (maxValue-minValue);
        this.origMinValue = minDistance * (maxValue-minValue);
    }

    private void initializeLattice(SOMLattice lattice){
        this.lattice.setLatticeHeight(lattice.getLatticeHeight());
        this.lattice.setLatticeWidth(lattice.getLatticeWidth());
        this.lattice.setNumberOfNodeElements(lattice.getNumberOfNodeElements());
```

```
        this.lattice.initializeValues();

        for (int i=0; i<this.lattice.getTotalNumberOfNodes(); i++){
            this.lattice.getLatticeNode()[i] = new Node(lattice.getLatticeNode()[i]);
        }

        this.lattice.setNodeHeight(lattice.getNodeHeight());
        this.lattice.setNodeWidth(lattice.getNodeWidth());
        this.lattice.setTotalNumberOfNodes(lattice.getTotalNumberOfNodes());
    }

    public SOMLattice getLattice() {
        return lattice;
    }

    private void initializeUmatrix(){

        double[] averageDistances = new double[lattice.getTotalNumberOfNodes()];
        double neighborhoodRadiusSquared = Math.pow((neighborhoodSize*Math.max(
            lattice.getNodeWidth(), lattice.getNodeHeight())), 2);
        maxDistance = 0;
        minDistance = 0;

        for(int i=0; i<lattice.getTotalNumberOfNodes(); i++){

            double distance = 0;
            double neighbors = 0;

            for(int j=0; j<lattice.getTotalNumberOfNodes(); j++){
                double distanceToNodeSquared = Math.pow((lattice.getLatticeNode()[i].
                    getXPos() - lattice.getLatticeNode()[j].getXPos()), 2) +
                                                Math.pow((lattice.getLatticeNode()[i
                                                    ].getYPos() - lattice.
                                                    getLatticeNode()[j].getYPos()),
                                                    2);

                if(distanceToNodeSquared <= neighborhoodRadiusSquared){
                    distance += lattice.getLatticeNode()[i].getDistance(lattice.
                        getLatticeNode()[j]);
                    neighbors++;
                }
            }
            averageDistances[i] = distance/(neighbors-1);

            if(i == 0){

                maxDistance = averageDistances[i];
                minDistance = averageDistances[i];
            }
            else{
                if(averageDistances[i]> maxDistance){
                    maxDistance = averageDistances[i];
                }

                if(averageDistances[i] < minDistance){
                    minDistance = averageDistances[i];
                }
            }
        }

        for(int i=0; i<lattice.getTotalNumberOfNodes(); i++){
            lattice.getLatticeNode()[i].setNodeColor(new Color((int)(255-(
                averageDistances[i]-minDistance)/(maxDistance-minDistance)*255),
                                                    (int)(255-(averageDistances[i]-
                                                        minDistance)/(maxDistance-
                                                        minDistance)*255),
                                                   (int)(255-(averageDistances[i
                                                       ]-minDistance)/(
```

```
                                                        maxDistance-minDistance)
                                                        *255)));

        }



    }

    @Override
    public void run() {
        //throw new UnsupportedOperationException("Not supported yet.");
        this.initializeUmatrix();
        try {
            Thread.sleep((int)Math.random()*10);
        } catch (InterruptedException ex) {
            Logger.getLogger(SOMTrainer.class.getName()).log(Level.SEVERE, null, ex);
        }

    }

    public void paintComponent(Graphics g){
        Graphics2D g2 = (Graphics2D)g;

        g.translate(this.getWidth()/2 - 225, this.getHeight()/2 - 240);
        for(int i=0; i<lattice.getTotalNumberOfNodes(); i++){

            g2.setColor(lattice.getLatticeNode()[i].getNodeColor());
            g2.fillRect(lattice.getLatticeNode()[i].getXPos() - lattice.getNodeWidth
                ()/2 ,lattice.getLatticeNode()[i].getYPos() - lattice.getNodeHeight()
                /2 ,lattice.getNodeWidth(),lattice.getNodeHeight());
            g2.setColor(Color.BLACK);
            g2.drawRect(lattice.getLatticeNode()[i].getXPos() - lattice.getNodeWidth
                ()/2 ,lattice.getLatticeNode()[i].getYPos() - lattice.getNodeHeight()
                /2 ,lattice.getNodeWidth(),lattice.getNodeHeight());

        }

        g.setColor(Color.BLACK);

        for(int i=0; i<inputData.getRows(); i++){
            g.drawString(inputData.getDataLabels()[i], lattice.getLatticeNode()[
                inputData.getInputDataBMUs()[i]].getXPos(), lattice.getLatticeNode()[
                inputData.getInputDataBMUs()[i]].getYPos());
        }

            double hund = 100;
            int intMaxValue = (int)(origMaxValue*hund);
            int intMinValue = (int)(origMinValue*hund);

            g.setColor(Color.BLACK);
            g.drawString(Double.toString(intMaxValue/hund), 0, 470);
            for(int i=0; i<256; i++){
                g.setColor(new Color(i,i,i));
                double width = 350;
                g.fillRect((int)Math.ceil(i*width/255) + 50, 455, (int)Math.ceil(
                    width/255), 15);
            }
            g.setColor(Color.BLACK);
            g.drawRect(50, 455, 350, 15);
            g.drawString(Double.toString(intMinValue/hund), 405, 470);
    }
}
```

SOMVisualize/Cluster.java

```
/*
 * To change lattice template, choose Tools | Templates
```

```
 * and open the template in the editor.
 */
package my.SOMVisualize;

import java.awt.Color;
import java.io.Serializable;
import java.util.logging.Level;
import java.util.logging.Logger;

/**
 *
 * @author Mark Lester
 */
public class Cluster extends SOMLattice implements Runnable, Serializable{

    private int numberOfClusters;
    private SOMLattice lattice;

    public Cluster(SOMLattice lattice, int numberOfClusters){

        this.numberOfClusters = numberOfClusters;
        this.lattice = new SOMLattice();
        initializeLattice(lattice);

    }

    public int getNumberOfClusters() {
        return numberOfClusters;
    }

    private void initializeLattice(SOMLattice lattice){
        this.lattice.setLatticeHeight(lattice.getLatticeHeight());
        this.lattice.setLatticeWidth(lattice.getLatticeWidth());
        this.lattice.setNumberOfNodeElements(lattice.getNumberOfNodeElements());

        this.lattice.initializeValues();

        for (int i=0; i<this.lattice.getTotalNumberOfNodes(); i++){
            this.lattice.getLatticeNode()[i] = new Node(lattice.getLatticeNode()[i]);
        }

        this.lattice.setNodeHeight(lattice.getNodeHeight());
        this.lattice.setNodeWidth(lattice.getNodeWidth());
        this.lattice.setTotalNumberOfNodes(lattice.getTotalNumberOfNodes());
    }

    public SOMLattice getLattice() {
        return lattice;
    }

    private void initializeCluster(){

        double[][] centroids = new double[this.numberOfClusters][lattice.
            getNumberOfNodeElements()];
        for(int i=0; i<this.numberOfClusters; i++){
            for(int j=0; j<lattice.getNumberOfNodeElements(); j++){
                centroids[i][j] = Math.random();
            }
        }

        boolean changeFlag = true;

        while(changeFlag){
            changeFlag = false;

            for(int i=0; i<lattice.getTotalNumberOfNodes(); i++){
                double[] distance = new double[this.numberOfClusters];
                double minDistance = 0;
```

```
                int clusterIndex = 0;

                for(int j=0; j<this.numberOfClusters; j++){
                    distance[j] = lattice.getLatticeNode()[i].getDistance(centroids[j
                        ]);
                }

                minDistance = distance[0];
                for(int j=1; j<this.numberOfClusters; j++){

                    if(distance[j] < minDistance){
                        minDistance = distance[j];
                        clusterIndex = j;
                    }

                }

                if(lattice.getLatticeNode()[i].getCluster() != clusterIndex){
            //      System.out.println(i +": "+lattice.getLatticeNode()[i].
                    getCluster() + " " + clusterIndex);
                    int clusterColor = 1024/this.numberOfClusters*clusterIndex;
                    Color nodeColor = new Color(0,0,0);
                    if(clusterColor < 256){
                        nodeColor = new Color(0,(clusterColor % 256),255);
                    }
                    else if(clusterColor < 512){
                        nodeColor = new Color(0,255,255 - (clusterColor % 256));
                    }
                    else if(clusterColor < 768){
                        nodeColor = new Color((clusterColor % 256),255,0);
                    }
                    else{
                        nodeColor = new Color(255,255 - (clusterColor % 256),0);
                    }
                    lattice.getLatticeNode()[i].setNodeColor(nodeColor);
                    lattice.getLatticeNode()[i].setCluster(clusterIndex);
                    changeFlag = true;
                }
            }

            centroids = new double[this.numberOfClusters][lattice.
                getNumberOfNodeElements()];
            int clusterCounter[] = new int[this.numberOfClusters];
            for(int i=0; i<lattice.getTotalNumberOfNodes(); i++){
                for(int j=0; j<lattice.getNumberOfNodeElements(); j++){
                    centroids[lattice.getLatticeNode()[i].getCluster()][j] += lattice
                        .getLatticeNode()[i].getDoubleElementAt(j);
                }
                clusterCounter[lattice.getLatticeNode()[i].getCluster()]++;
            }
            for(int i=0; i<this.numberOfClusters; i++){
                for(int j=0; j<lattice.getNumberOfNodeElements(); j++){
                    centroids[i][j] = centroids[i][j]/clusterCounter[i];
                }
            }
        }
    }

    @Override
    public void run() {
        this.initializeCluster();
        try {
            Thread.sleep((int)Math.random()*10);
        } catch (InterruptedException ex) {
            Logger.getLogger(Cluster.class.getName()).log(Level.SEVERE, null, ex);
        }
    }
}
```

SOMVisualize/Component.java

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package my.SOMVisualize;

import java.awt.Color;
import java.awt.Font;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.io.Serializable;
import javax.swing.*;

/**
 *
 * @author Mark Lester
 */
public class Component extends JPanel implements Serializable{

    private SOMLattice lattice;
    private int componentNumber;
    private double minValue;
    private double maxValue;

    private double origMaxValue;
    private double origMinValue;

    public Component(SOMLattice lattice, int componentNumber){
        this.lattice = new SOMLattice();
        this.componentNumber = componentNumber;

        initializeLattice(lattice);
        initializeComponentPlane();

    }

    private void initializeLattice(SOMLattice lattice){
        this.lattice.setLatticeHeight(lattice.getLatticeHeight());
        this.lattice.setLatticeWidth(lattice.getLatticeWidth());
        this.lattice.setNumberOfNodeElements(lattice.getNumberOfNodeElements());

        this.lattice.initializeValues();

        for (int i=0; i<this.lattice.getTotalNumberOfNodes(); i++){
            this.lattice.getLatticeNode()[i] = new Node(lattice.getLatticeNode()[i]);
        }

        this.lattice.setNodeHeight(lattice.getNodeHeight());
        this.lattice.setNodeWidth(lattice.getNodeWidth());
        this.lattice.setTotalNumberOfNodes(lattice.getTotalNumberOfNodes());
    }

    public SOMLattice getLattice() {
        return lattice;
    }

    private void initializeComponentPlane(){

        maxValue = lattice.getLatticeNode()[0].getDoubleElementAt(componentNumber);
        minValue = lattice.getLatticeNode()[0].getDoubleElementAt(componentNumber);

        for(int i=1; i<lattice.getTotalNumberOfNodes(); i++){

            if(lattice.getLatticeNode()[i].getDoubleElementAt(componentNumber) <
                minValue){
                minValue = lattice.getLatticeNode()[i].getDoubleElementAt(
                    componentNumber);
```

```
            }
            if(lattice.getLatticeNode()[i].getDoubleElementAt(componentNumber) >
                maxValue){
                maxValue = lattice.getLatticeNode()[i].getDoubleElementAt(
                    componentNumber);
            }
        }

        for(int i=0; i<lattice.getTotalNumberOfNodes(); i++){
            Node currNode = lattice.getLatticeNode()[i];
            int colorValue =  (int)Math.round((currNode.getDoubleElementAt(
                componentNumber) - minValue)/(maxValue-minValue)* 1020);
            Color nodeColor =  new Color(0);
            int caseValue = colorValue/256;

            if(caseValue == 0){
                nodeColor = new Color(0,(colorValue % 256),255);
            }
            else if(caseValue == 1){
                nodeColor = new Color(0,255,255 - (colorValue % 256));
            }
            else if(caseValue == 2){
                nodeColor = new Color((colorValue % 256),255,0);
            }
            else{
                nodeColor = new Color(255,255 - (colorValue % 256),0);
            }

            lattice.getLatticeNode()[i].setNodeColor(nodeColor);
        }
    }

    public double getMaxValue() {
        return maxValue;
    }

    public double getMinValue() {
        return minValue;
    }

    public void setOrigMaxMin(double maxValue, double minValue){
        this.origMaxValue = this.maxValue * (maxValue-minValue) + minValue;
        this.origMinValue = this.minValue * (maxValue-minValue) + minValue;
    }

    public void paintComponent(Graphics g){
        super.paintComponent(g);
        Graphics2D g2 = (Graphics2D)g;

        g2.scale(0.5, 0.5);

        for(int i=0; i<lattice.getTotalNumberOfNodes(); i++){

            g.setColor(lattice.getLatticeNode()[i].getNodeColor());
            g.fillRect(lattice.getLatticeNode()[i].getXPos() - lattice.getNodeWidth()
                /2 ,lattice.getLatticeNode()[i].getYPos() - lattice.getNodeHeight()/2
                 ,lattice.getNodeWidth(),lattice.getNodeHeight());
            g.setColor(Color.BLACK);
            g.drawRect(lattice.getLatticeNode()[i].getXPos() - lattice.getNodeWidth()
                /2 ,lattice.getLatticeNode()[i].getYPos() - lattice.getNodeHeight()/2
                 ,lattice.getNodeWidth(),lattice.getNodeHeight());

        }

        g.setColor(Color.BLACK);
        double hund = 100;
        int intMaxValue = (int)(origMaxValue*hund);
        int intMinValue = (int)(origMinValue*hund);
```

```
        g.setFont(new Font("Arial", Font.PLAIN, 18));
        g.drawString(Double.toString(intMinValue/hund), 0, 470);
        g.drawString(Double.toString(intMaxValue/hund), 405, 470);

        for(int i=0; i<1024; i++){
            if(i < 256){
                g.setColor(new Color(0,(i % 256),255));
            }
            else if(i < 512){
                g.setColor(new Color(0,255,255 - (i % 256)));
            }
            else if(i < 768){
                g.setColor(new Color((i % 256),255,0));
            }
            else{
                g.setColor(new Color(255,255 - (i % 256),0));
            }
            double width = 350;
            g.fillRect((int)Math.ceil(i*width/1024) + 50, 455, (int)Math.ceil(width
                /1024), 15);

        }
        g.setColor(Color.BLACK);
        g.drawRect(50, 455, 350, 15);
    }
}
```

SOMVisualize/MapBrowser.java

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package my.SOMVisualize;

import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.IOException;
import java.util.logging.Level;
import java.util.logging.Logger;


import java.awt.Canvas;
import java.awt.Dimension;
import java.awt.Toolkit;
import javax.swing.JFrame;

import ru.atomation.jbrowser.impl.JBrowserComponent;
import ru.atomation.jbrowser.impl.JBrowserBuilder;
import ru.atomation.jbrowser.impl.JBrowserCanvas;
import ru.atomation.jbrowser.impl.JComponentFactory;
import ru.atomation.jbrowser.interfaces.BrowserManager;

/**
 *
 * @author Mark Lester
 */
public class MapBrowser {

    private String mapUrl;
    private boolean isInitialized;
    private BrowserManager browserManager;


    public MapBrowser(String mapUrl){
        this.mapUrl = mapUrl;
        this.isInitialized = false;
```

```
}

public void runMap(){
    if(isInitialized){
        Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();

        JFrame frame = new JFrame();
        frame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        frame.setSize((int) (screenSize.getWidth() * 0.75f),
                (int) (screenSize.getHeight() * 0.75f));
        frame.setLocationRelativeTo(null);
        frame.setTitle("GIS Visualizer");

        if(browserManager == null){
        browserManager =
                new JBrowserBuilder().buildBrowserManager();
        }
        JComponentFactory<Canvas> canvasFactory = browserManager.
            getComponentFactory(JBrowserCanvas.class);
        JBrowserComponent<?> browser = canvasFactory.createBrowser();

        frame.getContentPane().add(browser.getComponent());
        frame.setVisible(true);

        browser.setUrl("http://localhost/SOMVisualize/gisVisualizer.php");
    }
}

public boolean initializeXML(Cluster clust, Component[] comp, InputData inputData
    ){
    FileWriter fw = null;
    BufferedWriter bw = null;

    try {
      fw = new FileWriter("temp.xml");
      bw = new BufferedWriter(fw);

      bw.write("<document>");
      bw.newLine();

      //write the labels
      bw.write("\t<labels>");
      bw.newLine();

      bw.write("\t\t<cluster>Clusters</cluster>");
      bw.newLine();

      for(int i=0; i<inputData.getCols(); i++){
          bw.write("\t\t<component" + i +">" + inputData.getVariableLabels()[i] +
                "</component" + i +">");
          bw.newLine();
      }

      bw.write("\t</labels>");
      bw.newLine();

      for(int i=0; i<inputData.getRows(); i++){

          //start of a block

          bw.write("\t<data>");
          bw.newLine();

          //write cluster name
          bw.write("\t\t<title>" + inputData.getDataLabels()[i] + "</title>");
          bw.newLine();

          //write cluster colors
```

```
                bw.write("\t\t<cluster>" + this.getHexRGB(clust.getLattice(), inputData
                    , i) +"</cluster>");
                bw.newLine();

                //write component colors
                for(int j=0; j<inputData.getCols(); j++){
                    bw.write("\t\t<component" + j +">" + this.getHexRGB(comp[j].
                        getLattice(), inputData, i) + "</component" + j +">");
                    bw.newLine();
                }

                //end block
                bw.write("\t</data>");
                bw.newLine();
            }

            bw.write("</document>");

            bw.close();

            isInitialized = true;
            return true;
        }
        catch (IOException e) {
            return false;
        }
    }

    private String getHexRGB(SOMLattice sl, InputData inputData, int i){
                String red = Integer.toHexString(sl.getLatticeNode()[inputData.
                    getInputDataBMUs()[i]].getNodeColor().getRed());
                String green = Integer.toHexString(sl.getLatticeNode()[inputData.
                    getInputDataBMUs()[i]].getNodeColor().getGreen());
                String blue  = Integer.toHexString(sl.getLatticeNode()[inputData.
                    getInputDataBMUs()[i]].getNodeColor().getBlue());

                if(red.length() == 1){
                    red = 0 + "" + red;
                }
                if(green.length() == 1){
                    green = 0 + "" + green;
                }
                if(blue.length() == 1){
                    blue = 0 + "" + blue;
                }

                return red+""+green+""+blue;
    }
}
```

SOMVisualize/InputData.java

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package my.SOMVisualize;

import java.io.Serializable;

/**
 *
 * @author Mark Lester
 */
public class InputData implements Serializable{
```

```
    private int rows;
    private int cols;

    private double[][] inputData;
    private int[] inputDataBMUs;
    private String[] dataLabels;
    private String[] variableLabels;

    public InputData(int rows, int cols){
        this.rows = rows;
        this.cols = cols;

        this.inputData = new double[rows][cols];
        this.inputDataBMUs = new int[rows];
        this.dataLabels = new String[rows];
        this.variableLabels = new String[cols];
    }

    public int getCols() {
        return cols;
    }

    public int getRows() {
        return rows;
    }

    public String[] getDataLabels() {
        return dataLabels;
    }

    public void setDataLabels(String[] dataLabels) {
        System.arraycopy(dataLabels, 0, this.dataLabels, 0, rows);
    }

    public double[][] getInputData() {
        return inputData;
    }

    public void setInputData(double[][] inputData) {
        for(int i=0; i<inputData.length; i++){
            System.arraycopy(inputData[i], 0, this.inputData[i], 0, cols);
        }
    }

    public int[] getInputDataBMUs() {
        return inputDataBMUs;
    }

    public void setInputDataBMUs(int[] inputDataBMUs) {
        System.arraycopy(inputDataBMUs, 0, this.inputDataBMUs, 0, rows);
    }

    public String[] getVariableLabels() {
        return variableLabels;
    }

    public void setVariableLabels(String[] variableLabels) {
        System.arraycopy(variableLabels, 0, this.variableLabels, 0, cols);
    }

    public void setOneBMU(int dataIndex, int BMUindex){
        this.inputDataBMUs[dataIndex] = BMUindex;
    }
}
```

SOMVisualize/InputFacility.java

```
package my.SOMVisualize;
```

```
import java.awt.Component;
import java.awt.event.*;
import java.io.Serializable;
import javax.swing.*;
import javax.swing.event.CellEditorListener;
import javax.swing.event.TableModelEvent;
import javax.swing.event.TableModelListener;
import javax.swing.table.*;


/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

/*
 * InputFacility.java
 *
 * Created on Mar 1, 2012, 9:05:02 AM
 */
/**
 *
 * @author Mark Lester
 */
public class InputFacility extends javax.swing.JFrame implements Serializable{

    private boolean ready;

    /** Creates new form InputFacility */
    public InputFacility() {
        initComponents();
        initTable();
    }

    /** This method is called from within the constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is
     * always regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-BEGIN:
        initComponents
    private void initComponents() {

        addColumn = new javax.swing.JButton();
        addRow = new javax.swing.JButton();
        jScrollPane1 = new javax.swing.JScrollPane();
        table = new javax.swing.JTable();
        jScrollPane2 = new javax.swing.JScrollPane();
        colTitleTable = new javax.swing.JTable();
        submitButton = new javax.swing.JButton();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

        addColumn.setText("Add Column");
        addColumn.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                addColumnActionPerformed(evt);
            }
        });

        addRow.setText("Add Row");
        addRow.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                addRowActionPerformed(evt);
            }
```

```
});

jScrollPane1.setPreferredSize(new java.awt.Dimension(300, 200));

table.setModel(new javax.swing.table.DefaultTableModel(
    new Object [][] {

    },
    new String [] {

    }
));
table.setAutoResizeMode(javax.swing.JTable.AUTO_RESIZE_OFF);
jScrollPane1.setViewportView(table);

colTitleTable.setModel(new javax.swing.table.DefaultTableModel(
    new Object [][] {

    },
    new String [] {
        "Column Number", "Column Title"
    }
) {
    boolean[] canEdit = new boolean [] {
        false, true
    };

    public boolean isCellEditable(int rowIndex, int columnIndex) {
        return canEdit [columnIndex];
    }
});
jScrollPane2.setViewportView(colTitleTable);

submitButton.setText("Submit");
submitButton.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        submitButtonActionPerformed(evt);
    }
});

javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane()
    );
getContentPane().setLayout(layout);
layout.setHorizontalGroup(
    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(layout.createSequentialGroup()
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.
            Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addComponent(addColumn)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.
                    RELATED)
                .addComponent(addRow)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.
                    RELATED)
                .addComponent(submitButton))
            .addGroup(layout.createSequentialGroup()
                .addContainerGap()
                .addComponent(jScrollPane2, javax.swing.GroupLayout.
                    PREFERRED_SIZE, 217, javax.swing.GroupLayout.
                    PREFERRED_SIZE)
                .addGap(18, 18, 18)
                .addComponent(jScrollPane1, javax.swing.GroupLayout.
                    PREFERRED_SIZE, 555, javax.swing.GroupLayout.
                    PREFERRED_SIZE)))
        .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.
            MAX_VALUE))
);
```

```
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.
                    Alignment.BASELINE)
                    .addComponent(addColumn)
                    .addComponent(addRow)
                    .addComponent(submitButton))
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.
                    Alignment.LEADING)
                    .addComponent(jScrollPane2, javax.swing.GroupLayout.DEFAULT_SIZE,
                        480, Short.MAX_VALUE)
                    .addComponent(jScrollPane1, javax.swing.GroupLayout.DEFAULT_SIZE,
                        480, Short.MAX_VALUE))
                .addContainerGap())
        );

        pack();
    }// </editor-fold>//GEN-END:initComponents

private void addRowActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:
    event_addRowActionPerformed
    Object[] dataArray = new Object[model.getColumnCount()];
    model.addRow(dataArray);


}//GEN-LAST:event_addRowActionPerformed

private void addColumnActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:
    event_addColumnActionPerformed
   colTitleTableModel.addRow(new Object[]{model.getColumnCount(), "Column" + model.
       getColumnCount()});
   model.addColumn("Column" + model.getColumnCount());

}//GEN-LAST:event_addColumnActionPerformed

private void submitButtonActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST
    :event_submitButtonActionPerformed
    submitForm();
}//GEN-LAST:event_submitButtonActionPerformed

private void initTable(){

    jScrollPane1.setBorder(BorderFactory.createTitledBorder("Data"));
    jScrollPane2.setBorder(BorderFactory.createTitledBorder("Column Headers"));

    model = (DefaultTableModel)table.getModel();
    colTitleTableModel = (DefaultTableModel)colTitleTable.getModel();

    colTitleTableModel.addTableModelListener(new TableModelListener(){

            @Override
            public void tableChanged(TableModelEvent e) {
                if(e.getType() == TableModelEvent.UPDATE){
                    String[] columnHeaders = new String[colTitleTableModel.
                        getRowCount()];

                    for(int i=0; i<colTitleTableModel.getRowCount(); i++){
                        columnHeaders[i] = colTitleTableModel.getValueAt(i, 1).
                            toString();
                    }
                    model.setColumnIdentifiers(columnHeaders);
                }
            }

    });
    this.setVisible(true);
```

```
        this.setTitle("SOM Visualize - Manual Data Input");
        this.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
    }

    private void submitForm(){
       ready = true;
       this.dispose();
    }

    public boolean isReady() {
        return ready;
    }

    public DefaultTableModel getModel(){
       return model;
    }
        // Variables declaration - do not modify//GEN-BEGIN:variables
        private javax.swing.JButton addColumn;
        private javax.swing.JButton addRow;
        private javax.swing.JTable colTitleTable;
        private javax.swing.JScrollPane jScrollPane1;
        private javax.swing.JScrollPane jScrollPane2;
        private javax.swing.JButton submitButton;
        private javax.swing.JTable table;
        // End of variables declaration//GEN-END:variables
        private DefaultTableModel model;
        private DefaultTableModel colTitleTableModel;
    }
```

SOMVisualize/ConsoleLog.java

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package my.SOMVisualize;

import java.awt.Color;
import java.io.Serializable;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.text.BadLocationException;
import javax.swing.text.StyleConstants;

/**
 *
 * @author Mark Lester
 */
public class ConsoleLog implements Serializable{
    public javax.swing.JTextPane log;
    private javax.swing.text.StyledDocument doc;

    public ConsoleLog(javax.swing.JTextPane log, javax.swing.text.StyledDocument doc){
        this.log = log;
        this.doc = this.log.getStyledDocument();

        //Set different styles
        javax.swing.text.Style style = this.log.addStyle("Success", null);
        StyleConstants.setForeground(style ,new Color(101,157,50));

        style = this.log.addStyle("Error", null);
        StyleConstants.setForeground(style, Color.red);

        style = this.log.addStyle("Normal", null);
        StyleConstants.setForeground(style, Color.black);
    }

    public void throwSuccessMessage(String s){
```

```
    try {
        doc.insertString(doc.getLength(), s, doc.getStyle("Success"));
        log.setCaretPosition(doc.getLength());
    }
    catch (BadLocationException ex) {
        Logger.getLogger(SOMVisualizeUI.class.getName()).log(Level.SEVERE, null, ex);
    }
}


public void throwErrorMessage(String s){
    try {
        doc.insertString(doc.getLength(), s, doc.getStyle("Error"));
        log.setCaretPosition(doc.getLength());
    }
    catch (BadLocationException ex) {
        Logger.getLogger(SOMVisualizeUI.class.getName()).log(Level.SEVERE, null, ex);
    }
}

public void throwNormalMessage(String s){
    try {
        doc.insertString(doc.getLength(), s, doc.getStyle("Normal"));
        log.setCaretPosition(doc.getLength());
    }
    catch (BadLocationException ex) {
        Logger.getLogger(SOMVisualizeUI.class.getName()).log(Level.SEVERE, null, ex);
    }
}
}
```

SOMVisualize/gisVisualizer.php

```
<?php include "./xml2array.php" ?>
<!DOCTYPE html>
<html>
    <head>
        <meta name="viewport" content="initial-scale=1.0, user-scalable=no" />
        <style type="text/css">
            html { height: 100% }
            body { height: 100%; margin: 0; padding: 0 }
        </style>
        <script type="text/javascript"
            src="http://maps.googleapis.com/maps/api/js?key=
                AIzaSyDMs8fkBNCpeM9fOS4YECJN_m0kOVRDDTg&sensor=false">
        </script>
    </head>
    <body onload="initialize()">
        <table width="100%" height="100%" id="mainTable">
            <tbody>
                <tr valign="top">
                    <td id="inputPanel" width="30%">
                        <?php
                            if($_POST['xmlTxt']){

                        ?>

                            <input type="hidden" id="hiddenXML" value="<?php echo(
                                $_POST['xmlTxt']); ?>" />
                            <input type="hidden" id="color" value="<?php echo($_POST
                                ['gisVis']); ?>" />

                        <?php
                            }
                        ?>
                        <?php
                            include "./inputPanel.php";
```

```
                    ?>

                </td>
                <td width="70%">
                    <div id="map_canvas" style="width: 100%; height: 800px"></div
                        >
                </td>
            </tr>
        </tbody>
    </table>
</body>

<!-- This part loads the map, do not alter -->
<script type="text/javascript">
    var map;
    var txt;
    var data;
    var xmlDoc;
    var componentCount;
    var read = false;

    function initialize() {
        //geocoder = new google.maps.Geocoder();
        var myOptions = {
          center: new google.maps.LatLng(13.521479136545901, 122.13656582812496),
          zoom: 6,
          mapTypeId: google.maps.MapTypeId.ROADMAP
        };
        map = new google.maps.Map(document.getElementById("map_canvas"),
            myOptions);

        //document.getElementById('files').addEventListener('change',
            parseXMLFile, false);

        google.maps.event.addListener(map, 'click', function(event) {
            var myLatLng = event.latLng;
            var lat = myLatLng.lat();
            var lng = myLatLng.lng();

            alert(lat + ", " + lng);

        });

        <?php
            if($_POST['xmlTxt']){
                //echo("txt = document.getElementById('hiddenXML').value;\n");
                echo("document.getElementById('radio' +" . $_POST['gisVis'] . ").
                    checked = true;");
                echo("visualize();\n");

            }
        ?>
    }
</script>
</html>
```

SOMVisualize/inputPanel.php

```
<table width="100%" height="100%" id="inputTable" valign="top" border="1" >
    <tbody id="inputBody">
        <tr height="10%" valign="top">
            <td id="fileInputRow" colspan="2" >
            <form id="fileInputForm" action="" method="POST" enctype="multipart/form-
                data">
                    <input type="file" id="fileInput" name="fileInput" onchange="
                        submit()" />
            </form>
            </td>
```

```
        </tr>
        <?php
            if($_FILES['fileInput'] || $_POST['xmlTxt']){
                if($_FILES['fileInput']){
                    $fileData = file_get_contents($_FILES['fileInput']['tmp_name']);
                    $xml_array = xml2array($fileData);
                }
                else{
                    $fileData = $_POST['xmlTxt'];
                    $xml_array = xml2array($_POST['xmlTxt']);
                }
                echo("<form name='toDisplay' action='' method='POST'>");
                echo("<tr bgcolor='#4EEE94'>
                    <td width='15%'>
                        <input type='radio' name='gisVis' id='radio1' value='1'
                            onclick='submit()'>
                    </td>
                    <td width='85%'>" .
                        $xml_array['document']['labels']['cluster'] .
                        "
                    </td>
                    </tr>");

                    for($i=0; $i<count($xml_array['document']['labels'])-1; $i++){
                        if($i%2 == 1){
                            echo("<tr bgcolor='#4EEE94'>");
                        }
                        else{
                            echo("<tr>");
                        }
                        echo("
                        <td width='15%'>
                        <input type='radio' name='gisVis' id='radio" . ($i+2) . "'
                            value='" . ($i+2) . "' onclick='submit()'>
                    </td>
                    <td width='85%'>" .
                        $xml_array['document']['labels']['component' . $i] .
                        "
                    </td>
                    </tr>");


                    }
                    echo("<tr>");
                    echo("<td colspan='2' align='center'>");

                    echo("<input type='hidden' value='$fileData' name='xmlTxt'
                        onclick='alert(this.id)'>");
                    echo("</td>");
                    echo("</tr>");
                echo("</form>");
            }
        ?>

    </tbody>
</table>

<script type="text/javascript">

    function visualize(){
        data = new Array();
        alert("Hi!");

        <?php
            for($i=0; $i<count($xml_array['document']['data']); $i++){
                echo("data[$i] = new Array(". count($xml_array['document']['data'][$i
                    ]) .");");
                for($j=0; $j<count($xml_array['document']['data'][$i]); $j++){
```

```
            if($j==0){
                echo("data[$i][$j] = '" . $xml_array['document']['data'][$i
                    ]['title'] ."';");
            }
            else if($j==1){
                echo("data[$i][$j] = '" . $xml_array['document']['data'][$i
                    ]['cluster'] ."';");
            }
            else{
                echo("data[$i][$j] = '" . $xml_array['document']['data'][$i
                    ]['component'.($j-2)] ."';");
            }
        }
    }
?>

var latLngArray;
var color;

<?php
    mysql_connect("localhost","root","") or die('Error connecting to mysql')
        ;
    mysql_select_db("gisvisualization") or die("Error");

    $myArray = xml2array($_POST['xmlTxt']);

    for($i=0; $i<count($myArray['document']['data']); $i++){
        $placeName = $myArray['document']['data'][$i]['title'];

        $query = "SELECT place_id from place WHERE place_name = '$placeName'"
            ;
        $data = mysql_query($query);
        $data = mysql_fetch_assoc($data);
        $placeId = $data['place_id'];

        $query = "SELECT polygon_id, place_latitute, place_longitude FROM
            place_latlng WHERE place_id='$placeId' ORDER BY polygon_id,
            latlng_id";
        $data = mysql_query($query);

        $latLngArray = array();

?>


        latLngArray = new Array(new Array());
        color = document.getElementById('color').value;



<?php
        while($result = mysql_fetch_row($data)){
            $latLngArray[$result[0]][] = "$result[1], $result[2]";
        }

        for($j=0; $j<count($latLngArray); $j++){
            echo("latLngArray[" . $j . "] = new Array();");
            for($k=0; $k<count($latLngArray[$j]); $k++){
                echo("latLngArray[" . $j . "][" . $k . "]" . " = new google.
                    maps.LatLng(" . $latLngArray[$j][$k] . ");\n");
            }
        }
?>

for(var i=0; i<latLngArray.length; i++){

   var polygon = new google.maps.Polygon({
        paths: latLngArray[i],
```

```
                strokeColor: "#" + data[<?php echo($i); ?>][color],
                strokeOpacity: 0.8,
                strokeWeight: 2,
                fillColor: "#" + data[<?php echo($i); ?>][color],
                fillOpacity: 0.5

            });

                var tooltip = document.createElement('div');
                tooltip.innerHTML = "<?php echo($placeName); ?>";

                google.maps.event.addListener(polygon,'mouseover',function(){
                    tooltip.style.visibility = 'visible';
                });
                google.maps.event.addListener(polygon,'mouseout',function(){
                    tooltip.style.visibility = 'hidden';
                });

            polygon.setMap(map);
        }

        <?php
            }
        ?>
    }


</script>
```

SOMVisualize/xml2array.php

```
<?php
/**
 * xml2array() will convert the given XML text to an array in the XML structure.
 * Link: http://www.bin-co.com/php/scripts/xml2array/
 * Arguments : $contents - The XML text
 *                $get_attributes - 1 or 0. If this is 1 the function will get the
     attributes as well as the tag values - this results in a different array
     structure in the return value.
 *                $priority - Can be 'tag' or 'attribute'. This will change the way
     the resulting array sturcture. For 'tag', the tags are given more importance.
 * Return: The parsed XML in an array form. Use print_r() to see the resulting array
     structure.
 * Examples: $array =  xml2array(file_get_contents('feed.xml'));
 *              $array =  xml2array(file_get_contents('feed.xml', 1, 'attribute'));
 */
function xml2array ($contents, $get_attributes=1, $priority = 'tag') {
    if(!$contents) return array();

    if(!function_exists('xml_parser_create')) {
        //print "'xml_parser_create()' function not found!";
        return array();
    }

    //Get the XML parser of PHP - PHP must have this module for the parser to work
    $parser = xml_parser_create('');
    xml_parser_set_option($parser, XML_OPTION_TARGET_ENCODING, "UTF-8"); # http://
        minutillo.com/steve/weblog/2004/6/17/php-xml-and-character-encodings-a-tale-
        of-sadness-rage-and-data-loss
    xml_parser_set_option($parser, XML_OPTION_CASE_FOLDING, 0);
    xml_parser_set_option($parser, XML_OPTION_SKIP_WHITE, 1);
    xml_parse_into_struct($parser, trim($contents), $xml_values);
    xml_parser_free($parser);

    if(!$xml_values) return;//Hmm...

    //Initializations
```

```
$xml_array = array();
$parents = array();
$opened_tags = array();
$arr = array();

$current = &$xml_array; //Refference

//Go through the tags.
$repeated_tag_index = array();//Multiple tags with same name will be turned into
    an array
foreach($xml_values as $data) {
    unset($attributes,$value);//Remove existing values, or there will be trouble

    //This command will extract these variables into the foreach scope
    // tag(string), type(string), level(int), attributes(array).
    extract($data);//We could use the array by itself, but this cooler.

    $result = array();
    $attributes_data = array();

    if(isset($value)) {
        if($priority == 'tag') $result = $value;
        else $result['value'] = $value; //Put the value in a assoc array if we
            are in the 'Attribute' mode
    }

    //Set the attributes too.
    if(isset($attributes) and $get_attributes) {
        foreach($attributes as $attr => $val) {
            if($priority == 'tag') $attributes_data[$attr] = $val;
            else $result['attr'][$attr] = $val; //Set all the attributes in a
                array called 'attr'
        }
    }

    //See tag status and do the needed.
    if($type == "open") {//The starting of the tag '<tag>'
        $parent[$level-1] = &$current;
        if(!is_array($current) or (!in_array($tag, array_keys($current)))) { //
            Insert New tag
            $current[$tag] = $result;
            if($attributes_data) $current[$tag. '_attr'] = $attributes_data;
            $repeated_tag_index[$tag.'_'.$level] = 1;

            $current = &$current[$tag];

        } else { //There was another element with the same tag name

            if(isset($current[$tag][0])) {//If there is a 0th element it is
                already an array
                $current[$tag][$repeated_tag_index[$tag.'_'.$level]] = $result;
                $repeated_tag_index[$tag.'_'.$level]++;
            } else {//This section will make the value an array if multiple tags
                with the same name appear together
                $current[$tag] = array($current[$tag],$result);//This will
                    combine the existing item and the new item together to make
                    an array
                $repeated_tag_index[$tag.'_'.$level] = 2;

                if(isset($current[$tag.'_attr'])) { //The attribute of the last(0
                    th) tag must be moved as well
                    $current[$tag]['0_attr'] = $current[$tag.'_attr'];
                    unset($current[$tag.'_attr']);
                }

            }
            $last_item_index = $repeated_tag_index[$tag.'_'.$level]-1;
            $current = &$current[$tag][$last_item_index];
```

```
        }

    } elseif($type == "complete") { //Tags that ends in 1 line '<tag />'
        //See if the key is already taken.
        if(!isset($current[$tag])) { //New Key
            $current[$tag] = $result;
            $repeated_tag_index[$tag.'_'.$level] = 1;
            if($priority == 'tag' and $attributes_data) $current[$tag. '_attr'] =
                $attributes_data;

        } else { //If taken, put all things inside a list(array)
            if(isset($current[$tag][0]) and is_array($current[$tag])) {//If it is
                already an array...

                // ...push the new element into that array.
                $current[$tag][$repeated_tag_index[$tag.'_'.$level]] = $result;

                if($priority == 'tag' and $get_attributes and $attributes_data) {
                    $current[$tag][$repeated_tag_index[$tag.'_'.$level] . '_attr'
                        ] = $attributes_data;
                }
                $repeated_tag_index[$tag.'_'.$level]++;

            } else { //If it is not an array...
                $current[$tag] = array($current[$tag],$result); //...Make it an
                    array using using the existing value and the new value
                $repeated_tag_index[$tag.'_'.$level] = 1;
                if($priority == 'tag' and $get_attributes) {
                    if(isset($current[$tag.'_attr'])) { //The attribute of the
                        last(0th) tag must be moved as well

                        $current[$tag]['0_attr'] = $current[$tag.'_attr'];
                        unset($current[$tag.'_attr']);
                    }

                    if($attributes_data) {
                        $current[$tag][$repeated_tag_index[$tag.'_'.$level] . '
                            _attr'] = $attributes_data;
                    }
                }
                $repeated_tag_index[$tag.'_'.$level]++; //0 and 1 index is
                    already taken
            }
        }

    } elseif($type == 'close') { //End of tag '</tag>'
        $current = &$parent[$level-1];
    }
}
return($xml_array);
}
```

XI. Acknowledgement

This special problem has been successfully completed through the help of several people. And I would like to thank each of them.

First of all, God, for giving me all these wonderful people and support that I need when I needed it the most. Through Him and His ways, I was able to conquer all the obstacles that have existed in the way.

To my mother, my motivation in every thing I do, thanks for making me feel the real life. Because of you, I have been really determined to complete this special problem and graduate on time. Thanks for all the love that you have given me. Thanks for raising me properly, my only parent.

To my adviser, Sir Geoff, thanks for being my father/adviser. For the topic you have given me and for trying to reach out to the experts for me. Thank you very much. You have indeed helped in making me understand the concepts that I needed to understand.

To my co-adviser/statisticians, Ma'am Iris and Ma'am Pia, thanks for making me realize the significance of my study to the field of computational statistics. For trying to help me see the applications of my study, for the test cases, for the coffee, the chismis and all. Thank you. Somehow, you have been my inspiration through all these. Sabay tayong g-graduate this year! Hurray! Big hug to the two of you! Thank you. :)

To my blockmates, thanks for being my companions for four straight years.

To Renzy, Beverly, Chessell, Vienna, and Lalay, thanks for being the first friends that I have made during my college life. It has been really memorable. Though we are parting ways, I hope we still see each other from time to time and have a friendly dinner. I will miss you guys. So much. :)

To Mommy JJ, Auradee, Trixe, thanks for helping me bring out my skills as a student. Kahit na hindi ako gumraduate as Cum Laude, at least, I have proven to myself that I am not inferior compared to other people. Thanks for being my groupmates, reviewmates, schedmates, lahat na. Thank you. :')

To my DPSM family, Ma'am Ji, Ma'am Fatsy, Ma'am Astrid, Sir Noel, Sir Yev, thanks for being my ultimate supporters. For continuously asking how my thesis is going, and

when I'll defend. Tita A, and Ate She for being there when I needed to rant. Because of you guys, I am able to express my disappointments and regrets. Thank you very much. :) To the rest of the people in DPSM, thank you for accepting me as your Student Assistant though I may have been very inefficient this semester. Thank you DPSM! I love you all! :)

Special mention for Ma'am Sarj, who tried her best in all her ways to support us in doing these requirements. For lending your house for one day. For staying up with me and trying to argue over several matters that are deemed needed for the bugs that I have encountered. For printing this manuscript and most of all for being a friend/mother in school. Thank you very much.

To my special someone, thanks for all the love you have given. For the continuous support that you have given, for Starbucks-ing with me each time I needed to concentrate and finish my SP. Thanks for trying your best to understand what I have been explaining that are very technical. Thanks for all the food, the movies, the TV shows that we watch together. I hope we last forever and continuously be my friend, my parent, my teacher, and everything else that you may be. I love you. :)

To my high school classmates, for the continuous pressure that you have given me by posting your graduation pictures. You have really affected the way I managed my time and effort. Thank you very much. Guian, thanks for being at the other side of the campus even if we don't meet that much. Thanks for answering my inquiries regarding the significance of my SP to your field. Thank you guys. :)

Lastly, to the staff of Starbucks Power Station (Macapagal), thank you for serving the coffee and food I need with a smile. For letting me stay long inside the coffee shop. Thanks for the perfect blend of "Grande Caramel Macchiato for Mark". Thank you. :)